TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

VALTTERI LAAKSONEN

SÄHKÖISTEN TENTTIEN AUTOMAATTINEN TARKASTAMINEN

Diplomityö

Tarkastajat: Simo Ali-Löytty, Tapio Elomaa
Tarkastajat ja aihe hyväksytty
09.08.2017

TIIVISTELMÄ

VALTTERI LAAKSONEN: Sähköisten tenttien automaattinen tarkastaminen
Tampereen teknillinen yliopisto
Diplomityö, 47 sivua, 9 liitesivua
syyskuu 2018
Teknis-luonnontieteellinen koulutusohjelma
Pääaine: Matematiikka
Tarkastajat: Simo Ali-Löytty, Tapio Elomaa
Avainsanat: sähköinen arviointi, sähköinen tentti, tentin tarkastus, automaattinen tarkastus

Sähköiset tentit ovat yleistyneet huomattavasti ja samalla näille on luotu monia eri tenttijärjestelmiä. Myös automaattisen tarkastamisen kehittäminen on hiljalleen kasvattanut suosiotaan. Suuri osa tästä kehityksestä on painottunut esseiden tarkastamiseen. Samaa kehitystä voitaneen myös soveltaa muissa tenttityypeissä.

Tässä työssä tarkastellaan eri tenttityyppejä ja miten niitä on sähköistetty, eli vaihdettu paperi vastausalustana tietokoneeseen. Esseiden kirjoittamiseen tietokonetta on jo käytetty pitkään, mutta tenttien tekemiseen sitä on alettu vasta viimeaikoina hyödyntämään suuremmalla mittakaavalla. Matematiikan tenttien sähköistämisessä suurin ongelma on matematiikan kaavojen ja merkkien kirjoittaminen tietokoneella, mutta tämä ei ole ylitsepääsemätön este.

Tietokoneiden hyödyntäminen vain tentteihin vastaamiseen jättää paljon tietokoneen potentiaalista käyttämättä; tietokoneilla voidaan myös tarkastaa tentit automaattisesti. Näin opettajien työtaakka pienenee huomattavasti, ja opiskelijoiden mahdollisuudet itsenäiseen oppimiseen kasvaa. Jo yksinkertainenkin ohjelma, joka helpottaa tenttijärjestelmistä saatavaa vastausaineiston hallintaa ja tarkastamista, johtaa tähän tavoitteeseen. Tässä työssä esitellään tähän tehtävään tehty ohjelma, jolle on annettu nimeksi PunaKynä.

ABSTRACT

VALTTERI LAAKSONEN: Automatic assessment of elecrtical exams
Tampere University of Technology
Master's thesis, 47 pages, 9 Appendix pages
September 2018
Master's Degree Programme in Science and Engineering
Major: Mathematics
Examiner: Simo Ali-Löytty, Tapio Elomaa
Keywords: electrical exam, electrical assesment, exam grading, automatic grading

Electrical exams have widespread considerably, and many different examination systems have been created for these. Even the development of automated grading has gained more popularity. Much of the development has focused on grading essays, but the same advances can be applied for other types of exams.

This thesis examines different types of exams, and how they have been computerised, or how the paper has been changed for a computer screen. Computers have been in use for writing essays for a long time, but only recently they have been used on a large scale for exams. The greatest problem with computerising mathematics' exams is writing mathematical formulas and symbols with computers, but this is not an insurmountable hurdle.

Utilising computers for just answering to exams leaves a lot of computational potential wasted, as computers could be used for grading the exams as well. This would considerably lighten the workload of teachers, and increase the students possibilities for self studying. Even a simple program to ease the managing and assessing of the answer corpus from an examination system would be a good step towards this goal. This thesis introduces a program that has been made for that task, named PunaKynä.

ALKUSANAT

Tämä diplomityö tehtiin Tampereen teknillisen yliopiston matematiikan laboratoriolle 2017-2018 välisenä aikana. Huhtikuun puolessa välissä vuonna 2017 alkoi PunaKynä-ohjelman kehittäminen, joka jatkui kiivaana yli kesän. Syksyllä 2017 PunaKynän kehitys hidastui, ja painottui lähinnä virheiden korjaamiseen. Tämän työn kirjoittaminen alkoi kesällä, jatkui pitkin lukuvuotta, ja saapui päätökseen seuraavana syksynä. Kirjoittamisen ohella pidin myös laskuharjoituksia ja toimin yleisesti tutkimusapulaisena matematiikan laboratoriossa.

Tyyliini kuuluen pidemmittä puheitta kiitän ohjaajiani Simo Ali-Löyttyä ja Tapio Elomaata tuesta, avusta, ja hitaan etenemiseni sietämisestä.

Tampereella, 18.9.2018

Valtteri Laaksonen

SISÄLLYS

LYHENTEET JA MERKINNÄT

BETSY	(Bayesian Essay Test Scoring sYstem)
CAS	Tietokoneen algebrajärjestelmä (Computer Algebra System)
EXAM	Sähköisten tenttien ohjelmisto
GPL	Yleinen julkinen lisenssi (General Public License)
HTML	Hypertekstin merkkauskieli (HyperText Markup Language)
IEA	(Intelligent Essay Assessor)
IEMS	(Intelligent Essay Marking System)
LaTeX	Tieteelliseen kirjoittamiseen soveltuva ladontajärjestelmä
LGPL	Lievempi yleinen julkinen lisenssi (Lesser General Public License)
LSA	Piilevän semantiikan analyysi (Latent Semantic Analysis)
LSI	Piilevän semantiikan indeksöinti (Latent Semantic Analysis)
LSTM	Pitkä lyhytaika muisti (hermoverkko) (Long Short-Term Memory)
MathCheck	Matemaattisten tehtävien tekemiseen ja vastausten tarkistamiseen luotu ohjelmisto
Matlab	Numeeriseen laskentaan tarkoitettu tietokoneohjelmisto.
Maxima	Matemaattiseen laskentaan käytetty CAS-ohjelmointikieli
MLE	Matlabin tieteellisen tekstin ja matemaattisten käskyjen ladonta ohjelma (Matlab Live Editor)
Moodle	Modular Object-Oriented Dynamic Learning Environment
NLP	Luonnollisen kielen käsittely (Natural Language Processing)
OCR	Optinen hahmontunnistus (Optical Character Recognition)
PDF	Liikutettava dokumenttimuoto (Portable Document Format)
PEG	(Project Essay Grade)
Qt	C++ ohjelmointikieleen perustuva ohjelmistojen kehitysympäristö
STACK	A System for Teaching and Assessment using a Computer algebra Kernel

a	Vakio
A	Matriisi
a_{ij}	Matriisin rivin i sarakkeen j alkio
A^T	Matriisin transpoosi
$\mathbb{R}^{m \times n}$	Reaalisten matriisien, joiden koko on $m \times n$, joukko
σ_i	Singulaariarvo
λ_i	Ominaisarvo
$\mathbf{u}$	Satunnaismuuttujavektori
$\mathbf{u}_i$	Satunnaismuuttujavektorin i. alkio
$\mathcal{A}$	Tapahtuma
$P(\mathcal{A})$	Tapahtuman $\mathcal{A}$ todennäköisyys
$P(\mathcal{A}\|\mathcal{B})$	Tapahtuman $\mathcal{A}$ todennäköisyys tapahtuman $\mathcal{B}$ ehdolla
$P_k(\mathcal{A})$	Tapahtuman $\mathcal{A}$ todennäköisyysketjun k. arvo
$B_{a \in C}$	Totuusarvo ominaisuuden a löytymiselle vastauksesta C

1. JOHDANTO

Tenttiminen on osa yliopiston arkea. Melkein jokainen suurelle joukolle järjestettävä kurssi loppuu tenttiin, ja se on yleisin kurssien suoritusvaatimus. Joidenkin kurssien aikana tehdään myös viikottain tehtäviä, jotka palautetaan kurssin pitäjille tarkastettavaksi. Joillakin kursseilla, varsinkin tietotekniikassa, järjestetään yksi tai useampi isompi harjoitustyö, jotka täytyy myös tarkastaa. Kurssien järjestämiseen kuuluu paljon työtä jota monikaan opiskelija ei välttämättä pysty tiedostamaan.

Tietokoneet ovat taasen osa jokapäiväistä arkea, niin kotona, työpaikoilla, kouluissa kuin yliopistoissakin. Suuri osa työstä ja viihteestä on integroituna tietokoneisiin. Opetuksessa niiden käyttöönotto on alkanut kuitenkin suhteellisen myöhään vasta viimeaikoina. Tietokoneita on myös alettu käyttämään tentteihin vastaamiseen. Tähän tehtävään tietokoneet avaavat uusia mahdollisuuksia tenttien laatuun ja tarkoitukseen; tietokonetta voi käyttää tenttitilaisuudessa muuhunkin kuin vain vastausten kirjoittamiseen.

Tietokoneiden käytön tenttien osalta ei tarvitse jäädä edes pelkästään tenttien tekemiseen ja niihin vastaamiseen. Tietokoneita voitaisiin myös käyttää tenttien välittömään automaattiseen tarkastamiseen. Kun tenttien tai harjoitusten tarkastamiseen kuluu vain muutama hetki, voidaan esimerkiksi harjoitustöitä järjestää kurssilla aiempaa useammin [8]. Esseetenttien kohdalla automaattista tarkastausta on jo alettu hyödyntämään. Matematiikan ja ohjelmoinnin tehtäville on olemassa ohjelmia, joilla voidaan vastauksia tarkistaa automaattisesti, mutta tenttien kohdalla tarkastaminen on vielä yleisesti ihmisen työtä.

Tässä työssä on tarkoitus pohtia, miten matematiikan tenttien arvostelua voitaisiin siirtää enemmän tietokoneen tehtäväksi. Tietyn tyyppisille, ei matemaattisille, tenteille on luotu useampiakin ratkaisuja arvostelun sähköistämiseksi, joten aluksi selvennetään hieman näiden toimintaperiaatteita. Mikäli valmiita menetelmiä voidaan soveltaa matematiikan tarpeisiin, niin niistä olisi hyvä lähteä liikkeelle. Matematiikan tehtävät noudattavat tosin hieman eri kaavoja kuin muiden aiheiden tehtävät, joten tämä on tuskin helposti tehtävissä. Jos tietokonetta ei voida valjastaa tarkistamaan tehtäviä itsenäisesti, niin ainakin sitä voidaan käyttää helpottamaan suuren

työn taakkaa.

Luvussa 2 tarkastellaan eri tyyppisiä tentti- ja harjoitustehtäviä, ja miten niitä arvostellaan. Tarkastelu kohdistuu monivalinta- ja esseekysymyksiin, sekä ohjelmoinnin ja matematiikan tehtäviin. Monivalintakysymykset ja ohjelmoinnin tehtävien tarkastelu työn osalta jää niiden esittämiseen. Nämä ovat jo nyt suhteellisen hyvin siirretty tietokoneiden tarkastettavaksi, koska ne soveltuvat siihen jo pohjaltaankin. Työssä keskitytään lähinnä esseekysymyksiin ja matemaattisiin tehtäviin. Esseekysymyksien tarkastamista on jo kehitelty kauan, ja kehitys jatkuu vieläkin. Matematiikan osalta kehitys on ollut heikompaa, ja usein on tyydytty vain tarkastamaan lopullinen vastaus tehtävistä, joissa yksinkertainen vastaus on mahdollista antaa.

Luvussa 3 perehdytään erilaisiin menetelmiin tarkastaa esseevastauksia automaattisesti tietokoneella. Luvun tarkoitus on kertoa hieman pintaa syvemmältä yleisimpien eri menetelmien ja mallien toiminnasta, sekä esitellä ohjelmia, jotka näitä menetelmiä käyttävät. Eri menetelmiä on esitettyjä enemmän, ja esiteltyjäkin menetelmiä käyttäviä ohjelmia on useampia. Esiteltävät menetelmät ovat LSA[11], NLP [39], Bayesin teoreema [31] ja hermoverkostot [36]. Menetelmien toiminta käydään pikaisesti läpi siten, että lukijalle jää suhteellisen selvä käsitys niiden toiminnasta. Hermoverkostoista voisi kirjoittaa kokonaan oman työn, joten niiden osalta läpikäynti on pinnallisempi. Lisäksi hermoverkostoja käyttäviä ohjelmia ei ainakaan yleisessä tiedossa näyttäisi olevan, vaikkakin niiden käyttö voi olla paras tapa luoda oppiva sähköinen arvostelija.

Esseekysymyksistä siirrytään matemaattisiin tehtäviin luvussa 4. Tässä luvussa esitellään, miten matematiikan tehtäviä on sähköistetty, ja miten niiden vastauksia voitaisiin myös tarkastaa automaattisesti. Luvussa esitellään kirjoittamisen aikana monella Tampereen teknillisen yliopiston kurssilla käytössä olevat Stack-tehtävät [34], MathCheck [46] ja Matlabin LiveEditor[44]. Näiden jälkeen pohditaan, miten matemaattisten tehtävien tarkastus sähköisesti voitaisiin suorittaa, ja testataan LSA- ja hermoverkkomenetelmiä Matlabin valmiiksi tarjoamilla kirjastoilla.

Luvussa 5 esitellään tämän työn ohella rakennettu PunaKynä ohjelma, jolla voidaan helpottaa sähköisten tenttien arvioimista, ja jonka päälle voidaan luoda automatisoitu tarkastusjärjestelmä. Luku käy läpi ohjelman päätoiminnot, ja ohjeistaa sen käytössä. PunaKynä on tehty Tampereen teknillisen yliopiston matematiikan laboratoriolle, ja se on tarkoitus lopulta julkaista yleisesti avoimen lähteen lisenssillä. PunaKynä on ollut jo käytössä muutamien kurssien tenttien tarkastamisessa, ja sen kehitys jatkuu vieläkin. PunaKynässä on jo alkukantainen automaattitarkistusominaisuus, joka toimii yksinkertaisella sanahaulla.

2. TENTTIEN AUTOMAATTINEN TARKASTUS

Iso osa opettajan työstä kuluu tenttien ja kokeiden parissa. Käytetystä ajasta suurin osuus menee niiden tarkastamisessa. Tehtävät, joissa vastaus on yksiselitteisesti joko oikein tai väärin, ovat nopeita tarkastaa, mutta tehtävät, joihin pyydetään pidempää tai täyttä esseevastausta, vaativat tarkastajilta huomattavasti enemmän aikaa ja vaivaa. Työtä voidaan jakaa useammalle opettajalle, mutta tällöin tarkastajien täytyy olla tarkkoja tarkastuskriteeriensä yhtäläisyydessä.

Monipuolisista tehtävistä on myös hyvä antaa monipuolista palautetta. Hyvä palaute auttaa opiskelijaa oppimaan, ja edesauttaa opiskelua. Paras palaute on jokaiselle opiskelijalle henkilökohtaisesti suunnattua ja opiskelijan tavoitteisiin suunniteltua.[15] Pienille ryhmille palautteen henkilökohtaistaminen on vielä realistisesti mahdollista, mutta ei enää useamman sadan kokoiselle ryhmälle. Suurissa ryhmissä yksilöt voidaan usen jakaa joukkoihin esimerkiksi vastauksien virhetyyppien mukaan, ja jakaa henkilökohtainen palaute joukkokohtaisesti. Tällöin palaute ei ole kaikista henkilökohtaisinta, mutta se voi silti olla rakentavaa.

Tenttien tarkastustyön siirtäminen tietokoneen tehtäväksi vapauttaa opettajien työaikaa huomattavasti. Pienellä alustustyöllä tuhansienkin tenttivastausten arviointi hoituu muutamassa hetkessä. Palautettakin tietokone voi antaa rakentavasti, jos sille on esimerkiksi annettu palauteteksti, jonka se antaa vastaukselle jossa se huomaa olevan tietynlainen virhe.

2.1 Monivalintakysymykset

Monivalintakysymykset koostuvat tehtävänannosta ja kahdesta tai useammasta vastausvaihtoehdosta, joista tentattavan on valittava yksi (tai useampi) rastittamalla tai mustaamalla pyydetty alue. Vastauksen tarkastaminen on täten helppoa ja nopeaa, koska tarkastajan täytyy vain verrata annetun vastauksen merkin paikkaa oikean vastauksen paikkaan. Tämän ihmistarkastaja tekee hetkessä, ja kone vielä nopeammin. Monivalintakysymyksiä voidaan toteuttaa tietokoneella tai paperilla. Paperille vastatut kysymykset saadaan helposti koneluettavaksi skannaamalla ne

optisella merkinnän lukijalla [2]. Skannatun paperin lukemiseen on useita, jopa vapaasti ladattavia ohjelmia [4].

Tarkastuksen nopeuden ansiosta monivalintatentit saavuttivat suuren suosion tietokoneiden kehityksen myötä, koska niillä voitiin nopeasti tentata suuria opiskelijajoukkoja lähes vaivatta [41]. Monivalintana vastauksen laajuus jää tosin vajaaksi siitä mitä tenttaamisella halutaan saavuttaa [45], ja tentattava on saattanut jopa vain arvata oikean vastauksen. Moniin määrällisiin tutkimuksiin monivalintakysymykset ovat tosin annetuista syistä suosittuja [40] ja soveltuvia, koska vastauksia tarvitaan hyvin monta, eikä vastausten ole yleensä tarkoituskaan tarkistaa oppimista.

2.2 Ohjelmoinnin tehtävien tarkastus

Ohjelmoinnin tehtävien, eli yleensä ohjelmakoodin tuottamisen, tarkastus ilman ohjelmaa on erittäin työlästä. Työn määrää opettajille on lisännyt myös tietotekniikan kasvanut kiinnostus opiskelijoiden keskuudessa, joka on näkynyt suuressa kirjautumismäärissä vastaaviin koulutusohjelmiin. Opettajien määrän suhteen opiskelijoiden määrään pienentyessä myös opettajien antamien tehtävien määrä vähenee niiden tarkastuksen viemän ajan takia. [8]

Ohjelmien tarkastamiseen sisältyy kolme eri kohtaa joihin täytyy kiinnittää huomiota. Tärkeysjärjestyksessä tärkeimmästä vähiten tärkeään ne ovat: toimivuus, tehokkuus ja työstettävyys [8]. Koodin toimivuus tarkoittaa, että sen pitää tuottaa kaikilla mahdollisilla syötteillä haluttu, oikea tai jopa väärä, lopputulos. Toimivuutta testataan siis antamalla ohjelmalle ennalta määritettyjä syötteitä, joiden oikea lopputulos on tiedossa, ja tätä verrataan ohjelman antamaan tulosteeseen. Tehokas koodi käyttää mahdollisimman vähän käytössä olevia resursseja, joita täytyy seurata ohjelman käydessä. Jotta koodi on työstettävää, sen pitää olla ihmiselle helposti ymmärrettävää.

Koodin toimivuuden ja tehokkuuden arviointi käsin on erittäin haastavaa. Koodi voi näyttää toimivalta ja tehokkaalta, mutta pienikin ero täysin toimivaan koodiin saattaa aiheuttaa virheitä ja muistivuotoja joillakin syötteillä. Lisäksi ohjelmoinnin ongelmiin on hyvin useasti useampia lähestymistapoja, ja erilaisia toimivia ratkaisuja on monia, jolloin tarkastajan täytyy lukea jokainen vastaus erittäin tarkasti. Näistä syistä ihmisarvioijat kiinnittävät usein enemmän huomiota koodin työstettävyyteen, jonka he pystyvät arvioimaan silmämääräisestikin. [8]

2.3 Esseetenttien tarkastus

Kiinnostavinta keskustelua tenttien automaattisesta tarkastamisesta käydään esseetenttien ympärillä. Voiko tietokone ymmärtää ihmisille ominaista luovuutta? Miten tietokone pystyisi koskaan antamaan yhtä rakentavaa palautetta kuin ihminen? Amerikkalaisen englannin opettajien neuvoston jyrkästä kritiikistä [22] huolimatta esseetenttien tarkastajien kehittely ja tutkiminen on jatkunut.

Esseetenttien tarkastamisen automatisointia on pyritty edistämään jo 1960-luvun lopulta lähtien, jolloin Page [24] ehdotti, että tietokoneet pystyisivät tarkastamaan myös esseitä, ja hieman myöhemmin julkaisikin PEG-ohjelman, joka teki tämän käytännössä. 1990-luvun alkaessa tietokoneet olivat kehittyneet ja yleistyneet tarpeeksi, että automaattinen tarkastaminen ja sen edelleen kehittäminen oli mahdollista ja käytännöllistä. Nykyään automaattiseen tarkastamiseen on olemassa monia kaupallisia ja avoimen lähdekoodin sovelluksia, jotka hyödyntävät erilaisia tarkastamisen malleja. Monen uudenkin tarkastusohjelman taustalla on vielä Pagen tekemä työ [48].

Esseetenttien tarkastus ei ole yksinkertainen prosessi. Pitkät kirjalliset vastaukset eroavat toisistaan huomattavasti, mutta täysin erilaisilla vastauksilla voidaan silti saavuttaa sama arvosana. Usein arvostelijat kuitenkin odottavat löytävänsä tiettyjä puheenaiheita tekstistä ennen kuin he ovat valmiita antamaan tälle täydet pisteet. Arvostelijoissakin on eroja, ja monet saattavat painottaa tekstin sujuvuuteen ja tyyliin enemmän kuin asiasisältöön [45].

Tämän monimuotoisen tarkastamisen siirtäminen tietokoneen hallintaan antaa näiden ongelmien lisäksi vielä omat haasteensa. Tietokoneen on vaikeampi arvioida vastauksen runollisempaa tyyliä, ja niin sanotun kapulakielen tunnistaminen vaatii erityistä huomiota, jos näihin halutaan tarkastuksessa panostaa. Kielestä riippuen sanamuodotkin tuottavat oman ongelmansa koneen algoritmeille.

Näistä vaikeuksista huolimatta automaattiseen tarkastukseen on luotu monia ohjelmia, joiden on myös todettu antavan hyvin lähelle samoja arvosanoja kuin ihmisarvostelijakin antaisi. Jokaisella ohjelmalla on tietty malli, jota ne käyttävät vastauksien tarkastamiseen. Mallit voivat tarkastaa vastauksesta tyylin ja sisällön, mutta yleensä painottuvat tarkastamaan vain jommankumman. Tutkimuksissa mallien tarkkuuksien välillä ei ole havaittu suuria eroja, riippumatta siitä tarkastavatko ne vain tyyliä vai vain sisältöä. [45]

Monien tämänhetkisten tarkistusmallien pohjalla on kokoelma ihmisten tarkistamia esseitä. Näistä malleista monet tarvitsevat kymmeniä, ellei satoja valmiiksi tar-

kastettuja esseitä muodostaakseen tarpeeksi tarkan arvostelukriteeristön. Joillekin ohjelmille voi myös antaa kurssimateriaalia kriteeristön luontiin [19], jolloin arvostelun tarkkuutta saadaan entistä paremmaksi. Arvostelukriteeristön luontiin voidaan myös käyttää vertailevaa arvostelua [27], jossa opettaja arvostelee muutamia vastauksia pareina, päättäen vain kumpi vastaus on parempi.

2.4 Matematiikan tenttien tarkastus

Matematiikan tenttien tarkastaminen on tietyiltä osin suoraviivaisempaa kuin vapaiden esseiden tarkastaminen, mutta luo myös erilaisia ongelmia. Matemaattisiin kysymyksiin on yleensä vain yksi vastaus, jonka oikeellisuus on helppo ja nopea tarkastaa. Vastauksilla voi kuitenkin olla useita muotoja, ja tämä täytyy ottaa huomioon tenttejä arvioitaessa ja luotaessa. Jos oppilasta on pyydetty antamaan vastaus tietyssä muodossa, voi tarkastaja jättää antamatta pisteen oikeasta, mutta väärän muotoisesta, vastauksesta. Jos vastaukselle ei ole annettu muotovaatimusta, on tarkastajalla suurempi vastuu päättää, minkä muotoisen vastauksen hän hyväksyy. Esimerkiksi jos tenttikysymyksenä on laskea luvun itseisarvo, niin vastauksessa ei todennäköisesti voida hyväksyä itseisarvomerkkejä.

Matematiikan tenteissä, niin kuin esseetenteissäkin, pelkkä vastauksen ilmaiseminen yhdellä luvulla tai lausahduksella ei usein riitä antamaan vastaukselle täysiä pisteitä. Vastaukset pitää myös yleensä perustella sanallisesti tai kaavoista johtamalla. Useasti pelkkä oikea perustelu on arvoltaan suurempi kuin oikea vastaus [50]. Tässä mielessä matematiikankin vastauksia voidaan tarkastella samoilla työkaluilla kuin esseevastauksia. Selvä ero esseevastaukseen on vastauksen, tai perustelun, pituus. Opiskelijoilta ei odoteta useita tekstikappaleita kirjoitettuja perusteluja, ja tämän vuoksi monet esseiden tarkastukseen soveltuvat menetelmät eivät sovi matematiikan tarkastamiseen ilman sovittelua.

Tähän asti matematiikan tehtävien automaattinen tarkastus on painottunut lähinnä vastauksen tai ennalta määritettyjen välivaiheiden oikeellisuuteen [34], ja viime aikoina jopa ennalta määräämättömien välivaiheiden tarkastamiseen [46]. Perustelujen arvostelu on huomattavasti haastavampi prosessi, ja matematiikan kirjoittaminen koneella on vaikeampaa ja hitaampaa kuin kynällä paperille. Asiakirjojen kirjoitusohjelmissa on omat komennot matematiikan symbolien kirjoittamiseen, LaTeX-tyyppisiä ohjelmistoja on olemassa, ja Matlab-ohjelmaan on tehty oma osio matematiikan kirjoittamiselle [44]. Silti moni oppilas toivoisi pystyvänsä ratkomaan tehtävät paperilla, vaikka lopullisen vastauksen kirjoittaisikin sitten koneella. [20]

3. ESSEETENTTIEN AUTOMAATTISEN TARKASTUKSEN RATKAISUJA

Esseiden tarkastuksen automatisointi alkoi 1960-luvulla, jolloin tietokoneet olivat harvinaisia ja niiden ohjelmat alkeellisia nykyisiin verrattuna. Tästä huolimatta Page onnistui luomaan ohjelman, joka pystyi tarkastamaan esseevastauksia lähes yhtä hyvin kuin joukko ihmistarkastajia [25]. Ajan ohjelmointirajoitteiden alaisena Pagen ohjelma, PEG, ei pystynyt jäsentämään tekstiä tarvittavalla tasolla, joten Page loi oman järjestelmänsä kiertämään tämän ongelman. Sen sijaan, että kone tutkisi tekstiä kokonaisuudessaan, se laski tekstistä asioita, mitä sen oli mahdollista laskea (joille Page antoi nimen 'prox'), ja käytti näitä arvoja arvioimaan haluttuja arvosteltavia osa-alueita (joille Page antoi nimen 'trins'). Esimerkiksi tekstin sujuvuus (trins) vastaa läheisesti sanojen määrää (prox), ja tyyli (trins) sanojen pituuksia (prox) [24]. Nyt esseevastauksia voidaan jäsentää tehokkaammin, ja Pagen proxit on mahdollista muuttaa entistä paremmin ja lähemmin arvioimaan trinsejä, ja joitakin trinsejä voidaan arvioida kokonaan ilman proxeja.

Kuten luvussa 2.3 mainittiin, nykyään on olemassa monia eri automaattitarkastajia, joiden toimintaperiaatteet eroavat toisistaan huomattavasti. Esseiden automaattitarkastajat voidaan jakaa neljään luokkaan niiden tarkastuskohteiden mukaan: koskeeko tarkastus kielellistä tyyliä vai aihesisältöä, ja mitataanko näitä välillisesti arvoa simuloivilla arvoilla vai arvoilla itsessään [48] (vrt. trins ja prox). Mitattavia arvoja kutsutaan yleisesti tekstin ominaisuuksiksi. Ohjelmat eivät ole rajoittuneet tarkastamaan vain tyyliä tai sisältöä, vaan voivat yhtä lailla tarkastaa molempia [45]. Taulukkoon 3.1 on koottu muutama esimerkkiohjelma niiden toiminta periaatteiden mukaan. Toimintaperiaatteet ja ohjelmat esitellään seuraavissa osissa.

	Sisältö	Tyyli
Simulaatio	IEA(LSA), BETSY(Bayes)	PEG(trins & prox), BETSY(Bayes)
Suora	ETS I(NLP), E-Rater(NLP)	E-Rater(NLP)

Taulukko 3.1 *Eri automaattitarkastusohjelmien sijoittuminen toimintaperiaatejakotaulukkoon [45]*

3.1 Piilevän semantiikan analyysi

Piilevän semantiikan analyysi, eli LSA, luotiin alunperin tiedonhaun tarpeisiin [47]. Tuolloin se tunnettiin nimellä piilevän semantiikan indeksöinti, LSI, joka laajensi aiempaa vektori-tyyppistä hakumenetelmää singulaariarvohajoitelmalla [26]. Singulaariarvohajotelma on matriisilaskennan menetelmä, jolla matriisi $A \in \mathbb{R}^{m \times n}$ jaetaan unitaarisiin matriiseihin $U \in \mathbb{R}^{m \times m}$ ja $V \in \mathbb{R}^{n \times n}$, sekä diagonaalimatriisiin $\Lambda \in \mathbb{R}^{m \times n}$ siten, että

$$A = U \Lambda V^T. \tag{3.1}$$

Olkoon r matriisin A nollasta eroavien singulaariarvojen määrä. Diagonaalimatriisi $\Lambda \in \mathbb{R}^{m \times n}$ koostuu alkioista a_{ij}, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ siten, että

$$a_{ij} = \begin{cases} \sigma_i & , i = j \wedge i \leq r \\ 0 & , i \neq j \vee i > r \end{cases},$$

missä $\sigma_i, i = 1, 2, \ldots, r$ ovat matriisin A nollasta eroavat singulaariarvot ja

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0.$$

Yhtälöä (3.1) kutsutaan matriisin A singulaariarvohajotelmaksi. Singulaariarvohajotelma voidaan tehdä kaikille matriiseille, jopa kompleksisille. Matriisit U ja V eivät ole yksikäsitteisiä, jos matriisilla A on moninkertaisia singulaariarvoja. [18]

Matriisin A singulaariarvot $\sigma_i > 0$ vastaavat matriisin $A^T A$ ominaisarvoja $\lambda_i > 0$ siten, että

$$\lambda_i = \sigma_i^2.$$

Vektorihaku, tai tiedonhaun vektoriavaruusmalli [10], luo siihen lisättyjen tekstien sanoista vektoriavaruuden, jossa perusajatus on, että jokainen eri sana vastaa yhtä ulottuvuutta. Sanojen taivutukset palautetaan yleensä alkeimuotoonsa, merkityksettömät sanat, kuten pronominit, voidaan jättää huomioimatta, ja ulottuvuuksiin voidaan myös sisältää kokonaisia lauseen pätkiä. Tekstit, joista avaruus on luotu, sijoitetaan avaruuteen vektoreina, joiden pituus jokaisella akselilla vastaa kyseisen akselin sanan löytymistä tekstistä. Pituudelle voidaan myös asettaa kertoimia esimerkiksi sen mukaan kuinka monta kertaa tai tiheästi sana esiintyy tekstissä, tai

kuinka tärkeä sana on holistisesti [3]. Painoarvoja voidaan myös muokata siten, että tekstien vektorit sijoittuvat avaruuteen mahdollisimman erilleen toisistaan [33]

Tekstejä haetaan avaruudesta hakusanoilla. Sanoista luodaan uusi pseudoteksti, joka sovitetaan avaruuteen vektoriksi samoin kuin muutkin tekstit. Hakuvektoria verrataan avaruuden muihin vektoreihin, ja tekstit, joiden vektorit ovat geometrisesti lähinnä hakuvektoria annetaan hakijalle. Tämä voidaan tehdä mm. laskemalla vektorien välisten kulmien kosinit. [3]

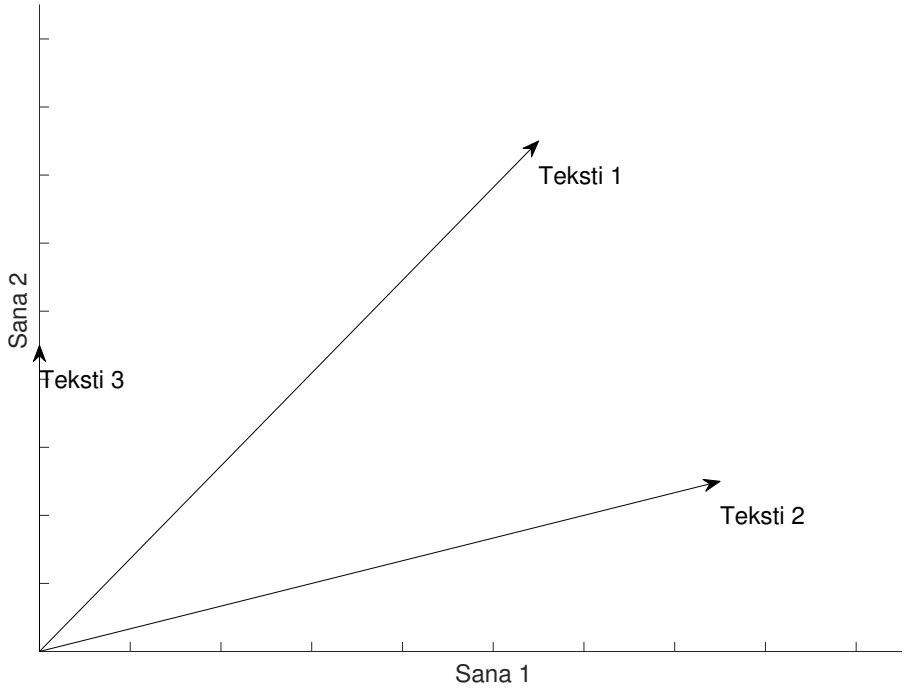
LSA muokkaa tekstien avaruutta singulaariarvohajotelmalla. Hajotetussa avaruudessa sanojen ja lausekkeiden välille syntyy yhteyksiä, joita tavallisessa vektoriavaruudessa ei ollut . Tekstin haku LSA-avaruuteen toimii mekaanisesti samalla tavalla kuin tavalliseen vektoriavaruuteen, mutta nyt yksittäinen hakusana virittääkin useampia ulottuvuuksia riippuen sen yhteyksistä muihin sanoihin. [11]

LSA-menetelmän tehokkuutta tavalliseen vektoripohjaiseen hakuun voidaan verrata hyvin kuvainnollisesti. Vektorimallinen haku etsii annettuja sanoja sellaisenaan tekstistä, luo vektoriavaruuden hakusanoista, ja määrittää tekstidokumenttien vektorit sanojen esiintymistiheyden mukaan [32]. Kuvaan 3.1 on piirretty kolmen eri tekstidokumentin vektoriesitys kahden hakusanan vektoriavaruuteen. Koska vektorimallin haku kohdistuu pelkästään sanoihin, sen ongelmana on muun muassa synonyymiset sanat. Olkoon haettava sana 1 "Matikka", ja sana 2 "Matematiikka". Vaikka sanat 1 ja 2 ovat käytännöllisesti samat, hakusanoina ne antavat eri tuloksia. Esimerkiksi kuvasta 3.1 nähdään, että sanalla "Matikka"hakumenetelmä ei löydä tekstiä 3.

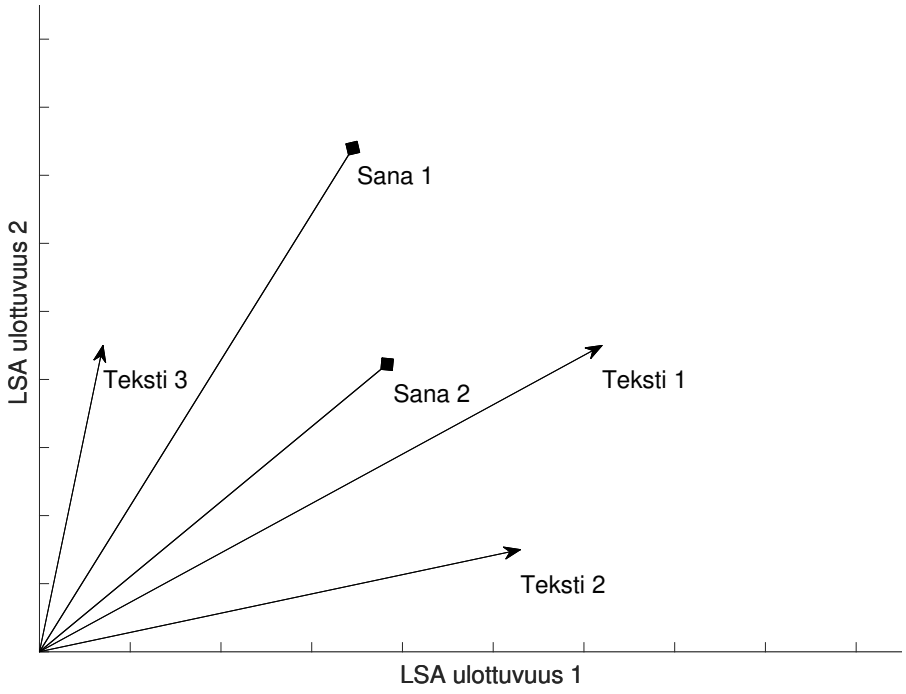
Jotta haku voidaan kohdistaa myös sanojen merkitykseen, sovelletaan haettaviin teksteihin LSA-menetelmää. Tutkittavista teksteistä luotavien matriisien jokainen rivi vastaa yksittäisiä sanoja, ja sarakkeet vastaavat joko koko tekstiä tai pieniä osia tekstistä, kuten kappaleita tai lauseita. Sarakeavaruuden valintaan vaikuttaa tutkittavan tekstin tyyppi ja tarkastelun tarkoitus. Matriisin alkiot vastaavat sanojen esiintymistaajuutta, usein muutettuna muotoon joka antaa parhaan tuloksen, kuten esimerkiksi taajuuden logaritmiin [11]. Saadulle matriisille tehdään tämän jälkeen singulaariarvohajotelma.

Saaduista singulaariarvoista valitaan tietty määrä suurimpia arvoja, ja loput voidaan asettaa nolliksi. Pienimpien singulaariarvojen voidaan olettaa vastaavan turhia sanoja ja muuta tilastollista kohinaa. Loput singulaariarvot ja niitä vastaavat vektorit luovat nyt tarkastusavaruuden, jota voidaan verrata muista teksteistä luotuihin avaruuksiin. [11]

Nyt kuvan 3.1 tekstit ja sanat voidaan piirtää LSA-menetelmästä saatujen ulottu-



Kuva 3.1 *Tekstidokumentit sanojen suhteen*



Kuva 3.2 *Tekstidokumentit ja sanat LSA-avaruudessa*

vuuksien mukaan kuvaan 3.2. Nyt, koska molemmat sanat esiintyvät samanlaisissa konteksteissa, sanalla "Matikka"voidaan löytää myös teksti 3. Vastaavasti voidaan tehdä muille sanoille, jotka esiintyvät samoissa yhteyksissä.

LSA-mallia on jatkokehitetty mm. todennäköisyys LSA-mallilla (Probabilistic La-

tent Semantic Analysis, PLSA) ja piilevän Dirichlet'n jaolla (Latend Diriclet Allocation, LDA). Vaikka nämä mallit ovat tehokkaampia tiedonhaussa, ne eivät kuitenkaan tehosta esseiden automaattista tarkastusta. [19]

3.1.1 Intelligent Essay Assessor -ohjelma

IEA [7] on automaattitarkastusohjelma, joka käyttää tarkastukseen LSA-mallia. LSA-avaruus luodaan tarkastettavan tehtävän aiheeseen kuuluvilla teksteillä, esimerkiksi oppikirjoilla ja muilla vastaavila. Mallia käytetään vertaamaan opiskelijan vastauksen sijaintia LSA-avaruudessa jo arvioitujen tekstien sijaintiin, ja arvioi vastausta vielä muutamilla muilla työkaluilla saavuttaakseen tarpeeksi tarkan arvion vastauksen pisteytykselle. Samalla vastausta tutkitaan plagiarisoinnin ja muiden huijausten varalta. [16]

Mallin pisteytyksen kalibrointiin riittää 20-100 esipisteytettyä tekstiä. Kalibrointi voidaan myös suorittaa asettamalla IEA kouluttamaan itsensä ilman esipisteytystä, tai vertaamalla vastauksia suoraan ideaalisiin vastauksiin tai oppikirjojen kappaleisiin. IEA voi vastausta pisteyttäessään myös osoittaa opiskelijalle kirjoja ja tekstejä joilla vastaus saisi paremman pisteen. [16]

3.2 Luonnollisen kielen käsittely

Luonnollinen kieli (Natural Language) tarkoittaa ihmisten tuottamaa puhetta ja tekstiä. Luonnollisen kielen prosessointi (NLP) sisältää erilaisia mekanismeja, joilla luonnollinen kieli pyritään muuttamaan muotoon, jonka tietokone pystyy ymmärtämään [39]. NLP menetelmiä käytetään erityisesti mm. konekääntäjissä ja puheentunnistusjärjestelmissä, [17] ja niitä on myös sovellettu esseiden automaattitarkastukseen.

Neljä tyypillistä NLP menetelmää on puheen osien merkkaus (part-of-speech tagging), paloittelu (chunking), nimellisen osan tunnistus (named entity recognition) ja semanttisen roolin nimeäminen (semantic role labeling). Puheen osien merkkaus pyrkii nimeämään sanojen roolit lauseessa, eli jakamaan lauseet niiden lauseenjäseniin. Paloittelu nimeää osia lauseesta syntaktisiin olennaisuuksiin, kuten substantiivi- tai verbi-ilmauksiin. Nimellisen osan tunnistus pyrkii tunnistamaan lauseista tunnettuja osia, kuten henkilöiden tai paikkojen nimiä, tai aikamääreitä. Semanttisen roolin nimeäminen antaa lauseen syntaktisille olennaisuuksille semanttisen roolin, kuten esimerkiksi 'predikaatin argumentti'. [9]

3.2.1 E-rater-ohjelma

E-rater on voittoa tavoittelemattoman Educational Testing Service -järjestön [38] luoma automaattinen esseiden tarkastusohjelma, joka käyttää tarkastamiseen NLP mekanismeja. Kirjoitus-, tyyli- ja mekaniikkavirheiden poimimiseen e-rater hyödyntää Criterion [6] -ohjelmaa, joka samalla antaa vastaajalle kirjallista palautetta virheistä. Criterion voi tarkistaa vastauksesta myös vaaditun rakenteen. Vastauksista lasketaan sanaston taso Brelandin standardisoidun taajuusindeksin (Breland's Standardized Frequency Index [5]) avulla, ainutlaatuisten sanojen määrän kaikkien sanojen suhteen, sanojen keskimääräisen pituuden ja sanojen lukumäärän. Näiden lisäksi e-rater laskee vielä aihekohtaisen sanaston käytön sisältövektorianalyysillä [33]. Lasketulle ominaisuuksille käytetään algoritmia, joka luo näistä regressiomallin, jonka avulla vastaukselle annetaan sen arvosana. [1]

E-raterin vanha versio 1.3, vaati vähintään 500 aihekohtaista ennalta tarkastettua esseetä luodakseen tarpeeksi tarkan arvosanakriteeristön. Versio 2.0 luo kriteeristön geneeriseltä taustalta, eikä vaadi tähän aihekohtaisia esseitä. Aihekohtaisuus vaikuttaa vain sen sanaston arvosteluun. Nämä menetelmät ovat osoittautuneet paljon tehokkaammiksi ja tuottavat jopa tarkempia arvosanoja. Testit ovat osoittaneet e-raterin antamien arvosanojen korreloivan ihmisten antamiin arvosanoihin 93% tarkkuudella. [1]

3.3 Bayesin teoreema

BETSY [30] on vapaasti ladattava automaattitarkastusohjelma, jonka Rudner rakensi käyttäen hyväksi Bayesin teoreemaa. Sen sijaan että tarkastaja antaisi vastaukselle suoraan tietyn arvosanan, BETSY laskee vastauksen mahdollisille arvosanoille todennäköisyydet. Vastaukselle annetaan sitten sen todennäköisin arvosana. Todennäköisyyksien laskemista varten määrilletään erilaisia ominaisuuksia, joita oletetaan esiintyvän tietyn arvosanan arvoisesta vastauksesta. Ominaisuudet voivat olla ennalta määritettyjä avainsanoja tai -sanajonoja, tai vaikka muilla menetelmillä (esim. LSA ja NLP) luotuja arvoja [31].

Bayesin teoreeman käyttö aloitetaan määrittelemällä todennäköisyydet $P(\mathbf{u}_k = 1|\mathcal{C})$ jokaiselle ominaisuudelle ja arvosanalle. Nyt $\mathbf{u}$ on kaikkien mahdollisten ominaisuuksien vektori, jossa $\mathbf{u}_k = 1$ tarkoittaa että ominaisuus k löytyy vastauksesta, ja $\mathcal{C}$ on tapahtuma sille, että jokin vastaus C on saanut arvosanan c. Tapahtumaa $\mathcal{C}$ voidaan merkitä myös $C = c$. Todennäköisyydet $P(\mathbf{u}_k = 1|\mathcal{C})$ voidaan määrittää esimerkiksi tutkimalla valmiiksi arvosteltuja vastauksia.

Teoreeman avulla lasketaan todennäköisyydet vastauksen arvosanoille $P_m(\mathcal{C})$, missä m on kaikkien tutkittavien ominaisuuksien lukumäärä. Todennäköisyyden $P_m(\mathcal{C})$ laskeminen suoritetaan ominaisuus kerrallaan siten, että todennäköisyys $P_{k+1}(\mathcal{C})$ lasketaan todennäköisyyden $P_k(\mathcal{C})$ avulla. Yleensä näitä todennäköisyyksiä ei indeksoida, vaan sanotaan, että todennäköisyys $P(\mathcal{C})$ päivitetään jokaisella ominaisuuden todennäköisyydellä. Tähän käytetään Bayesin teoreeman päivityskaavaa [31]

$$P(\mathcal{A}|\mathcal{B}) \cdot P(\mathcal{B}) = P(\mathcal{B}|\mathcal{A}) \cdot P(\mathcal{A}). \tag{3.2}$$

Jos tutkittavasta vastauksesta löytyy ominaisuus $\mathbf{u}_k$, niin $P(\mathbf{u}_k = 1) = 1$, ja kaavasta (3.2) saadaan sopivasti sijoittamalla

$$P_k(\mathcal{C}|\mathbf{u}_k = 1) = P_k(\mathbf{u}_k = 1|\mathcal{C}) \cdot P_{k-1}(\mathcal{C}). \tag{3.3}$$

Todennäköisyydet $P_k(\mathcal{C})$ lasketaan edellisten arvosanojen todennäköisyyksien keskiarvolla

$$P_k(\mathcal{C}) = \frac{P_{k-1}(\mathcal{C}|\mathbf{u}_i = 1)}{\sum_{n=1}^{m} P_{k-1}(C = n|\mathbf{u}_i = 1)}. \tag{3.4}$$

Ilman ennakkotietoa opiskelijan kyvyistä, voidaan todennäköisyydet $P_0(\mathcal{C})$ asettaa yhtä suuriksi. Nyt kaavoilla (3.3) ja (3.4) voidaan laskea kaikki todennäköisyydet $P_k(\mathcal{C})$, jokaiselle ominaisuudelle $k = 1, 2, \ldots, m$.

Bayesin teoreemaa voidaan käyttää useammalla tarkastusmallilla, jotka käyvät tarkastettavasta vastauksesta läpi joko kaikki mahdolliset tai tietyllä tavalla rajoitetun määrän ominaisuuksia. Yksi tekstin luokittelussa useimmiten käytetyistä malleista on Bernoullin monimuuttujamalli. Bernoullin malli käy läpi kaikki esimääritetyt ominaisuudet, ja tarkastaa sisältääkö tarkastettava vastaus ominaisuuden ainakin kerran. Tällä mallilla todennäköisyys, että vastaukselle D annetaan arvosana d, lasketaan kaavalla

$$P(\mathcal{D}) = \prod_{k=1}^{m} \left(B_{Dk} P(\mathbf{u}_k = 1|\mathcal{C}) + (1 - B_{Dk})(1 - P(\mathbf{u}_k = 1|\mathcal{D}))\right), \tag{3.5}$$

missä m on kaikkien ominaisuuksien lukumäärä ja $B_{Di} \in (0, 1)$ ilmaisee ominaisuuden i löytymistä vastauksesta D. Todennäköisyys $P(u_i = 1|\mathcal{D})$ saadaan laskettua

ennalta arvostelluista koulutusvastauksista kaavalla

$$P(u_k = 1|c) = \frac{1 + \sum_{j=1}^{T_d} B_{Aj}}{M + T_d}, \tag{3.6}$$

missä T_d on arvosanan d saaneiden koulutusvastausten lukumäärä. Nyt vastaukselle D voidaan antaa sen todennäköisin arvosana. [31]

Toinen yleisesti käytetty malli on monijäseninen malli, joka tarkastaa vain vastauksesta löytyvät ominaisuudet, ja niiden esiintymisen määrän. Tämä on laskennallisesti kevyempi ja usein parempi tarkastusmalli kuin Bernoullin malli, mutta ei välttämättä sovellu jokaiselle tarkastuskokonaisuudelle [31].

3.4 Hermoverkot ja syväoppiminen

Useat syväoppimismallit (deep learning) käyttävät hermoverkkojärjestelmiä. Hermoverkko (Neural Network) on koneoppimisen menetelmä, joka luo sisääntulon ja ulostulon välille solmuja (tai hermoja), jotka ovat yhteydessä toisiinsa tietyillä painoarvoilla [37]. Painoarvojen laskemiseen on monia tapoja [36], joista moneen kuuluu koulutuspaketti, joka sisältää eri sisääntuloja joiden halutut ulostulot tiedetään. Hermoverkko käy itsensä läpi useampia kertoja, optimoiden solmujensa välisi painoarvoja siten, että kaikki sisääntulot antaisivat ulostuloksi halutun arvon. Matalaoppimismalli (shallow learning) saadaan luotua latomalla useampi (erilainen) hermoverkko päällekkäin tasoiksi siten, että jokaisen tason ulostulo syöttää tietoa seuraavan tason sisääntuloon. Lisäämällä tasojen määrää saavutetaan syväoppimismalli [36]. Lukumäärärajaa näiden kahden mallin välillä ei ole yleisesti määritelty. Molempia voidaan yleisesti kutsua hermoverkostoiksi.

Hermoverkostoja käytetään laajalti esimerkiksi tekoälyissä ja kuvan- ja puheentunnistuksessa [43]. Niitä voidaan myös käyttää tekstien kategorisointiin [37], ja siten myös esseiden automaattiseen tarkastukseen. Tällöin hermoverkoston sisääntulona on tarkastettava tentti, ja ulostulona sen arvosana. Hermoverkot koulutetaan kuin aiemmin esitetyt tarkastusmenetelmät, eli mallille annetaan joukko valmiiksi arvosteltuja esseitä. Vuonna 2000 oli olemassa ainakin yksi automaattitarkastusohjelma, Intelligent Essay Marking System (IEMS), joka käytti tarkastamiseen hermoverkostoa [45], mutta tästä ei löytynyt tarkempaa tietoa, ja lähteenä annettua verkkosivua ei enää ole.

4. MATEMATIIKAN TARKASTUS

Matematiikan tehtävien tarkastukseen voidaan käyttää samoja työkaluja kuin esseetehtävien tarkistamiseen, mikäli tehtävässä vaditaan perusteluja. Oikea vastaus on tosin yleensä yksiselitteinen, ja erilaisia hyväksyttäviä perusteluja on rajoitettu määrä, eivätkä niiden pituudet ylitä edes useampaa kappaletta. Täsmälleen samojen ohjelmien ja algoritmien käyttö matematiikassa kuin essee vastauksissa ei tästä johtuen ole tehokasta resurssien käyttöä.

Matematiikan kirjoittamisessa täytyy huomata, että sanojen käsite on hieman monimutkaisempi kuin tavallisessa tekstissä. Luonnollisessa kielessä sanat erotetaan toisistaan tyhjällä merkillä (välilyönnillä). Matematiikan kielessä tyhjiä merkkejä käytetään lähinnä lukemisen helpottamiseksi, eikä niiden käyttö ole välttämätöntä ('$x = y$' ja '$x = y$' ovat matematiikan kirjoittamisen syntaksissa identtiset). Matemaattisesta vastauksesta täytyy pystyä poimimaan erilliseen jäsentelyyn luonnollinen ja matemaattinen syntaksi. Samalla täytyy myös huomioida, mitä osia matemaattisesta tekstilauseesta voidaan jakaa eri "sanoiksi".

Kirjoitusta vaikeuttaa myös matematiikan käyttämät symbolit. Monet kirjoitusjärjestelmät pystyvät kyllä esittämään eri aakkostojen, kuten kreikan aakkoston, merkkejä ilman ongelmia, mutta monet muut matematiikan käyttämät merkinnät ovat monimutkaisempia. Jotkut merkinnät tarvitsevat symbolien kirjoittamista eri tasolle kuin muu teksti (mm. potenssi ja jakoviiva), ja jotkut piirtämistä symbolien ylle tai alle (mm. juuri). Matematiikan ladontajärjestelmät piirtävät matemaattiset merkit halutusti, mutta vaativat käyttäjän kirjoittavat ne hallitulla syntaksilla. Järjestelmiä on useita, ja jokaisella on oma syntaksinsa, jota tarkastusta suorittavan järjestelmän täytyy osata tulkata.

Kirjoitettu matematiikka voidaan syöttää tietokoneelle tietokoneen algebrajärjestelmän (CAS) avulla. CAS pystyy käsittelemään sille syötetyistä teksteistä muuttujat erikseen, ilman että käyttäjän tarvitsee syöttää kaikki tieto numeerisesti. Funktioita voidaan näin myös vertailla eri muodoissa, ja sieventää tarvittuun muotoon. Algebrajärjestelmän avulla on mahdollista tarkastaa hyvin jäsenneltyä matemaattista vastausta vaikka sitä ei olisi kirjoitettu täsmälleen samoin kuin tarkastaja on

kirjoittanut. [42]

Matematiikan tehtävien tarkastaminen voidaan jakaa kahteen osaan: vastaus ja perustelu. Vastaus on joko lopullinen vastaus kysymykseen, tai koko rikkoutumaton päättelyketju siihen asti. Usein tenttien tehtävät vaativat myös perustelun, joka rakentuu vastauksen ympärille koostuen sanallisesta selityksistä, vastausta tukevista päättelyketjuista, tai molemmista.

4.1 Vastauksen tarkistus

Matematiikan tehtävien arviointia vastauksen oikeellisuuden suhteen on tehty jo pidemmän aikaa. Tähän on luotu ohjelmia, jotka antavat oppilaalle tehtävän, ja vertaavat oppilaan antamaa vastausta oikeaan vastaukseen. Tällaisien ohjelmien luominen on helppoa jos tehtävät ja vastaukset pidetään yksinkertaisina, mutta monimutkaisemmilla tehtävillä työn määrä kasvaa. Tehtävien arvostelu on yleensä binäärinen 'oikein/väärin', ja suurempaa piste-erottelua saadaan aikaiseksi laajentamalla yhden tehtävän koostumaan useammasta pienestä tehtävästä.

4.1.1 STACK-tehtävät

Monissa yliopistoissa ja koulutuslaitoksissa on käytössä selaimella käytettävä Moodle -oppimisalusta [21], jonka kautta opetuskurssien pitäjät voivat mm. jakaa opiskelijoille kaiken tarvittavan kurssimateriaalin, harjoituskysymykset ja niille palautussivun. Moodleen on myös luotu mahdollisuus tehdä kysymyksiä opiskelijoille. Kysymystyyppejä on monia, yksinkertaisista monivalintakysymyksistä esseekysymyksiin. Näihin liittyvä automaattinen tarkastus toimii tosin vain kysymyksissä, joihin on yksinkertainen ja ainutlaatuinen vastaus, joka voidaan tarkistaa suoraan vertaamalla oppilaan vastausta opettajan antamaan.

STACK-järjestelmän versio 3 on luotu ohjelmistoliitännäiseksi Moodle -ympäristölle, laajentamaan sen kysymystyyppejä matematiikan tarpeisiin. Opettajat voivat luoda monimutkaisiakin STACK -tehtäviä, jotka antavat opiskelijalle rakentavaa palautetta välittömästi. STACK ei vain vertaa annettua vastausta kirjain kirjaimelta oikeaan vastaukseen, vaan tutkii niiden matemaattista yhtäsuuruutta käyttäen Maxima CAS -ohjelmaa. Kysymyksien arvoja voi myös satunnaistaa, jolloin jokaisen opiskelijan kysymys voi olla erilainen. [34]

STACK-järjestelmän heikkous on sen käytettävyys. Monipuoliset tehtävät saadaan aikaiseksi siten, että kysymyksen tekijä kirjoittaa Maxima-syntaksista koodia, ja

opiskelija vastaa samalla syntaksilla. Tästä johtuen opettajien pitää opetella jonkin verran uutta ohjelmointikieltä, ja opiskelijoille pitää usein neuvoa, miten vastaus pitää kirjoittaa, jotta CAS pystyy lukemaan ja tarkastamaan vastauksen. Moodlen STACK-kysymysten luonnin käyttöliittymä on myös hieman hutera, koska sekin on vain HTML-sivu. Kysymysten rakenteen muuttaminen vaatii sivun uudelleen lataamisen, jolloin sivun koodi pyrkii poimimaan teksteihin kirjoitettuja määrättyjä muuttujia muokattaviksi. Lisäksi MathJax-koodin kirjoittaminen hankaloituu, kun HTML vaatii omia merkkejä kirjoitetun tekstin sisään, ja valitettavasti ohjelmointivirheen sattuessa sivu ei pysty kertomaan missä kyseinen virhe on.

4.1.2 MathCheck

MathCheck on Antti Valmarin kehittämä, ja vielä kehityksessä oleva CAS matemaattisen päättelyketjun tarkastamiseen [46]. Opiskelija voi syöttää sille useampia rivejä välivaiheita, joita MathCheck tarkistaa yksitellen, ja ilmoittaa missä kohdassa välivaihe ei enää vastaa lähtökohtaa. MathCheckin tavallisimpana käyttöliittymänä toimii HTML-sivu, jonka lomakkeeseen opiskelija syöttää vastauksensa. Kokenut internetsivujen tekijä pystyy siis tekemään tehtäväsivusta täysin omanlaisensa. [46] Mathcheckistä on myös komentoriviversio, jolle voidaan syöttää tarkastettavat vastaukset mistä vain käyttöliittymästä.

MathCheckin yleinen käyttöliittymä sisältää tehtävänannon, vastauslaatikon ja palautuspainikkeen. Vastattuaan tehtävään opiskelija antaa ratkaisunsa MathCheckille, joka käy ratkaisun rivi riviltä läpi, tarkistaen että jokainen rivi on matemaattisesti identtinen tehtävänannon kanssa. Vastaaja saa sitten palautteen oikeasta ratkaisusta, tai tiedon siitä, millä rivillä ratkaisu ei enää ollut oikein. Palaute voi olla myös huomautus syntaksivirheestä, eli jos vastaaja on syöttänyt MathCheckille tuntemattomia symboleja tai komentoja. Vastaukselle voidaan myös asettaa merkkimääräraja, jolla varmistetaan, että ratkaisu on tarkka tai riittävästi sievennetty.

Palautteen laatua voidaan myös rajoittaa. Esimerkiksi tenttitilaisuudessa ei välttämättä ole toivottua ilmoittaa opiskelijalle saman tien ratkaisun oikeellisuudesta. Tällöin vastauksen palauttaminen kirjoittaa ratkaisun palautettavaan lähdekoodiin tarkistamatta sitä tilaisuudessa, vaan opettaja antaa sen MathCheckille tarkastettavaksi tenttiä tarkastaessaan.

Niin kuin STACK -tehtävienkin kanssa, MathCheck vaatii hieman ohjelmointikäsitystä kysymysten laatijalta ja niihin vastaajalta. Verkkokäyttöliittymälläkin on hyvät ja huonot puolensa ohjelman käytön suhteen. MathCheck on vaivaton ottaa käyttöön, koska mitään ei tarvitse ladata koneille, ja tehtävien luonti vaatii

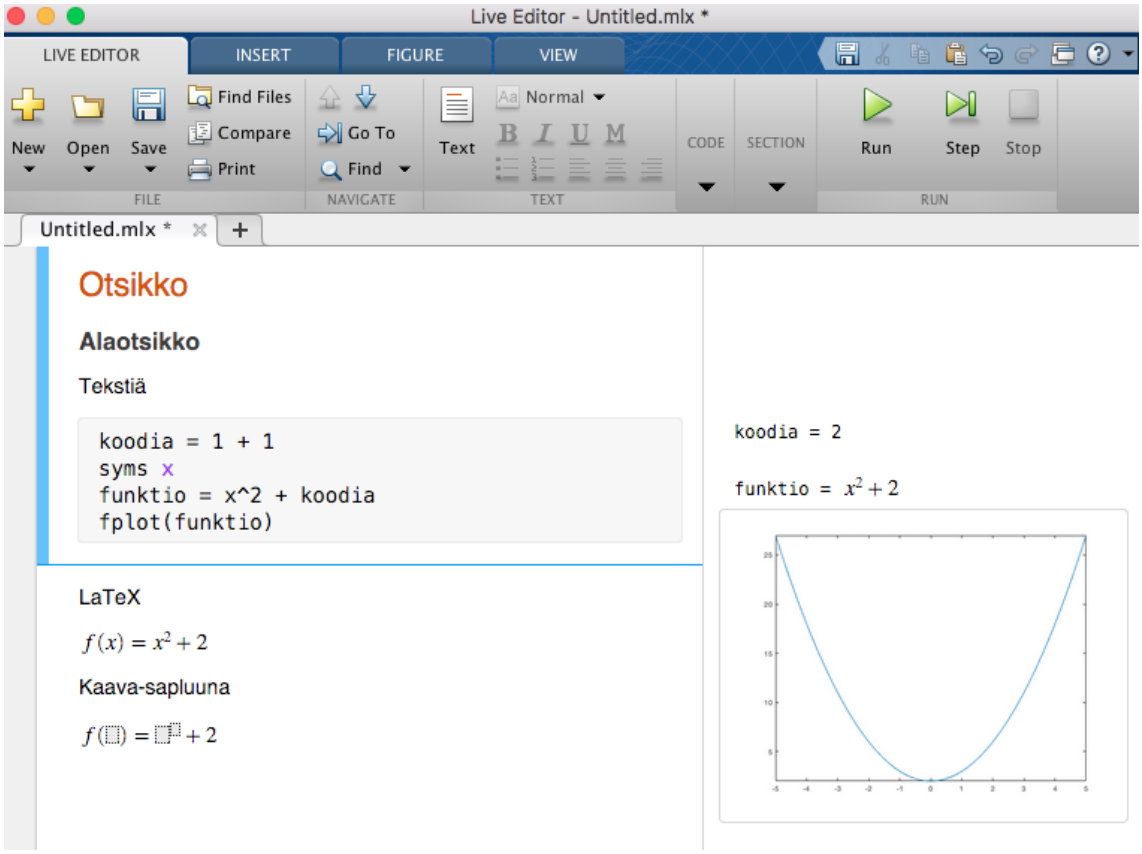
vain tekstieditorin ja paikan palvelimelta. Toisaalta ilman kokemusta internetsivujen tekemisestä tehtävien tekeminen voi koitua hyvin työlääksi. Tähän on tosin tehty MathCheck-tehtävien laadintaohjelma, jolla saa yksinkertaisen näköisen, mutta toimivan tehtäväsivun tehtyä. Ohjelman käyttö verkon kautta voi myös aiheuttaa palvelimen ylikuormitusta, josta voi aiheutua toimenpiteiden hidastumista.

4.2 Tarkastettavat sähköiset matematiikan tentit

Tenttien tekeminen koneella on viime aikoina yleistynyt huomattavasti. Ylioppilaslautakuntakin on päättänyt muuttaa ylioppilaskirjoitukset tietokoneilla tehtäviksi [49]. Esseetenttien suorittaminen näppäimistöllä tietokoneen näytölle ei teoriassa juurikaan eroa kynällä paperille kirjoittamisesta ja on käytännössä monella tapaa tehokkaampaa. Tenttien vastauksia on myös helpompi siirtää paikasta ja tarkastajalta toiselle, ja kuten kappaleessa 2 mainittiin, vastauksia pystytään myös tarkistamaan ja arvostelemaan koneellisesti.

Matematiikan osalta sähköiset tentit ovat hieman hankalampia toteuttaa. Moni tekstinkäsittelyohjelma tarjoaa matemaattisten kaavojen kirjoittamiseen oman muokkausosionsa, mutta tämä ei ole välttämättä riittävä. Joskus on toivottavaa lisätä vastaukseen myös kuvia. Ylioppilastutkintolautakunnan ratkaisu on antaa mahdollisuus lisätä kuvia kolmannen osapuolen ohjelmista kuvankaappauksella [51]. Tampereen teknillisessä yliopistossa päädyttiin käyttämään Matlabin Live Editor (MLE)-ominaisuutta [20].

MLE yhdistää yksinkertaisen tekstinkäsittelyohjelman, matemaattisen kaavakäsittelyohjelman ja Matlab koodausympäristön [44](kuva 4.1). Matemaattisten kaavojen kirjoittamiseen ohjelma tarjoaa valmiita kaavasapluunoita ja LaTeXkielen tulkin. Dokumenttiin kirjoitettu Matlab koodi on ajettavissa kuten tavallisessa Matlab -komentoikkunassa, ja sen tulokset tulostuvat samaan dokumenttiin. Valmis dokumentti voidaan tallentaa Matlabin omalla tiedostotunnisteella, tai se voidaan tulostaa PDF-tiedostoon. Tulostettu PDF-tiedosto vastaa siten käytännössä palautettua tenttipaperia.



Kuva 4.1 *MLE käyttöliittymä*

4.3 Vapaiden vastauksien tulkinta

Vaikka matematiikan tehtävien vastaus on yleensä yksiselitteinen, se ei aina ole yksinkertainen tai muodoltaan ainutlaatuinen. Sama pätee erityisesti virheille. Virheistä on hyvä antaa vastaajalle rakentavaa palautetta, jotta tämä voi ymmärtää virheensä ja kehittää oppiaan. Mahdollisia vääriä ratkaisuja on kuitenkin äärettömästi enemmän kuin oikeita, olettaen että tehtävällä on mielekäs ratkaisu. Ihmistarkastaja voi antaa jokaiselle vastaukselle yksilöllistä palautetta, mutta tietokone ei tähän pysty. Jos tietokone voisikin kategorisoida eri virheet niiden tyyppeihin, se voisi myös antaa rajoitettua virhekohtaista palautetta. Oikeat ratkaisut ansaitsevat yleensä muutenkin täydet pisteet (vastauksen osalta), eikä niistä tarvitse antaa yksilöllistä palautetta. Jos vastauksia on useampia erilaisia, ja niistä halutaan antaa eriäviä palautteita, voidaan niihin soveltaa samaa analyysiä kuin virheille.

Insinöörimatematiikka 123 -kurssin opiskelijat vastasivat MathCheckillä luotuun tenttiin, josta he tallensivat lähdekoodin, ja palauttivat sen tenttijärjestelmään vastauksenaan. Lähdekoodit käsiteltiin ohjelmalla, joka poimi palautetusta koodista tehtävänannon ja opiskelijan lopullisen vastauksen, ja syötti nämä uudestaan

MathCheckiin tarkistettavaksi. Näin tentin tarkastaminen voitiin automatisoida täysin. Tentissä oli 16 tehtävää, joista jokaisesta oli satunnaistettu 3 eri tehtäväpakettia, joko muuttamalla tehtävää tai vain tehtävänannon muuttujia. Jokaiseen tehtävään ei tarvinnut vastata. Tentin suoritti noin 400 opiskelijaa, ja tehtäväpaketit jakautuivat opiskelijoille hyvin tasaisesti.

Analysointia varten valittiin tehtävistä kaksi tehtävää, joihin oltiin saatu tarpeeksi vastauksia, ja joiden tehtävänannot olivat muuttujia lukuun ottamatta samat. Eri muuttujat vaikuttavat vain oikeaan ratkaisuun, josta ei tässä analyysissä olla kiinnostuneita. Virhetyypit ja niiden jakaumat ovat kuitenkin pitkälti samoja muuttujista huolimatta. Valitut tehtävät olivat tehtävät 1 ja 3. Tehtäväkohtaiset, keskenään täysin identtiset, vastaukset on taulukoitu liitteessä A.1 ja A.2.

Taulukkoon 4.1 on kerätty kaikki tehtävän 1 matemaattisesti identtiset väärät vastaukset. Vastauksien matemaattinen identtisyys tarkastettiin MathCheckillä. Sieventämättömät vastaukset pitäisi normaalisti käsitellä erikseen, sillä vaikka ne ovat matemaattisesti oikein, eivät ne välttämättä toteuta kysymykselle tarkoituksenomaisen vastauksen kriteerejä. Tässä tapauksessa kaikki syntaksivirheiset vastaukset ovat hyväksyttäviä, ja ne voidaan jättää huomioimatta virheellisten vastausten analyysissä.

Taulukosta 4.1 on helpompi luokitella vastauksia virhetyyppeihin. Tehtävän 1 vääristä vastauksista voidaan poimia esimerkiksi seuraavat virhetyypit: tyhjä vastaus, kielletty merkki, likiarvo ja laskuvirhe. Tyhjät vastaukset on helppo tunnistaa, ja niihin voidaan lukea myös vastaukset, joissa on pelkät sulkumerkit, mukaan lukien itseisarvomerkit. Kielletyt merkit tarkoittavat yleensä, että vastausta ei ole saavutettu: Tehtävässä 1 kielletyt merkit ovat itseisarvomerkit. Likiarvot voidaan tunnistaa vertaamalla vastausta oikeaan sallitulla virhemarginaalilla, ja loput vastaukset kuuluvat laskuvirheisiin. Joissain tilanteissa voi olla myös mielekästä jakaa sama vastaus useampaan kategoriaan, esimerkiksi väärin laskettu likiarvo kielletyllä merkillä. Tehtävän 1 virheet on luokiteltu taulukkoon 4.2.

Tehtävän 3 matemaattisesti identtiset väärät vastaukset on kirjattu taulukkoon 4.3. Tehtävän 3 sieventämättömät vastaukset voidaan jättää analyysissä huomioimatta samoin perustein kuin tehtävän 1 sieventämättömät vastaukset. Virheet voidaan myös luokitella samoin kuin tehtävässä 1. Tässä tehtävässä kiellettyjä merkkejä ovat "sin"ja "cos", sillä tehtävät on luotu niin, ettei vastaukseen tarvita trigonometrisiä funktioita. Tehtävän 3 virheet on luokiteltu taulukkoon 4.4.

Nyt väärille vastauksille voidaan antaa pisteet ja palautteet taulukon 4.2 kategorioiden mukaan. Virheitä voitaisiin myös jatkoanalysoida ennen pisteytystä, ja tutkia

Tehtävänanto	Laske $\lvert\sqrt{5}-3\rvert$		Laske $\lvert\sqrt{6}-4\rvert$	
	Vastaus	lkm.	Vastaus	lkm.
Väärä vastaus		15		7
	$\frac{7639}{10000}$	15	$4+\sqrt{6}$	7
	$3+\sqrt{5}$	11	$\frac{3101}{2000}$	5
	$-\lvert\sqrt{5}-3\rvert$	9	$\frac{31}{20}$	5
	$\lvert 5^{\frac{1}{2}}-3\rvert$	6	$\lvert\sqrt{2\cdot 3}-4\rvert$	3
	$(\frac{559}{250}-3)\cdot(-1)$	3	10	2
	$\frac{1}{2}$	3	$\sqrt{6}^2-4^2$	1
	$\frac{19}{25}$	3	$-\frac{155051}{100000}$	1
	$\sqrt{4}$	3	$4-\frac{1}{36}$	1
	22	2	6^2-4	1
	4	2	$\frac{1}{36}-4$	1
	$\lvert\rvert$	2	$\frac{2}{3}$	1
	-4	1	$\frac{1550510257}{10000000000}$	1
	$-(\sqrt{5}+3)$	1	$\sqrt{6}-4$	1
	1	1	$\sqrt{6}-\sqrt{2}$	1
	2.236068	1	x	1
	$\frac{3}{4}$	1	$\lvert 2\sqrt{3}-4\rvert$	1
	$\frac{5^1}{2}-3$	1		
	$\frac{763}{1000}$	1		
	$\sqrt{5-3}$	1		
	$\lvert-3\rvert$	1		
	0.763932	1		
Yhteensä	22	84	17	40

Taulukko 4.1 *Tehtävästä 1 poimitut opiskelijoiden väärät vastaukset*

Virhetyyppi	lkm.
Tyhjä vastaus	24
Kielletty merkki	17
Likiarvo	35
Laskuvirhe	48
Yhteensä	124

Taulukko 4.2 *Tehtävästä 1 poimitut opiskelijoiden väärät vastaukset kategorisoituna*

T.	Laske $\sin\frac{\pi}{4} - \sin\frac{7\cdot\pi}{4}$		Laske $\sin\frac{\pi}{4} - \sin\frac{3\cdot\pi}{4}$		Laske $\sin\frac{\pi}{4} - \sin\frac{7\cdot\pi}{4}$	
	Vastaus	lkm.	Vastaus	lkm.	Vastaus	lkm.
V.	$-\sin\frac{3\cdot\pi}{2}$	34	-1	22	$-\frac{\sin 6\cdot\pi}{4}$	34
	$-\sin 6\cdot\pi$	31		13		22
		27	1	4	$-\sin 3\frac{\pi}{2}$	20
	$-2\cdot\sin\frac{-3\cdot\pi}{4}$	4	$-\frac{1}{2}\cdot\pi$	3	$2\cdot\sin\frac{\pi}{4}$	5
	$\sin\frac{3\pi}{2}$	4	$-\sin\frac{\pi}{4}$	3	-1	3
	-6	3	$-\frac{2}{\sqrt{2}}$	2	$\frac{7071}{5000}$	3
	$\frac{7071}{5000}$	3	$\frac{2}{\sqrt{2}}$	2	$-\frac{6\pi}{4}$	2
	$\frac{2}{\sqrt{2}} - -\frac{2}{\sqrt{2}}$	2	$\sin\pi$	2	$1 - -1$	2
	-0.08209995	1	-180	1	$\frac{707}{1000} - -\frac{707}{1000}$	2
	$1 - -1$	1	$-\frac{5551}{5000}e - 16$	1	$-\frac{3}{2}$	1
	$2\cdot\cos\pi\cdot\sin -2\cdot\frac{\pi}{3}$	1	$-\pi$	1	$-\sin\frac{\pi}{4}$	1
	$3\cdot\frac{\pi}{2}$	1	$1 - -1$	1	$2\cdot\sqrt{2}$	1
	$=$	1	$2\cdot\frac{\pi}{4}$	1	$45 - 7\cdot 45$	1
	$\frac{7017}{10000} -$	1	2π	1	$\frac{1}{2}$	1
	$\frac{7071}{10000}$	1	3	1	$\frac{1}{2} -$	1
	$\sin\frac{1}{\pi}$	1	$=$	1	$\frac{2\cdot\sqrt{2}}{4}$	1
			$\frac{2}{4}$	1	$\frac{\pi}{2}$	1
					$\frac{19}{200}$	1
					π	1
					$\sin\frac{1}{7}$	1
					$\sin\frac{1}{\sqrt{2}} - \sin 7\cdot\frac{1}{\sqrt{2}}$	1
					$\sin\frac{\pi}{4} - 7\cdot\sin\frac{\pi}{4}$	1
					$\sqrt{2} - 2\cdot\sqrt{2}$	1
					x	1
Y.	16	116	17	60	24	108

Taulukko 4.3 *Tehtävästä 3 poimitut opiskelijoiden väärät vastaukset*

Virhetyyppi	lkm.
Tyhjä vastaus	64
Kielletty merkki	142
Likiarvo	9
Laskuvirhe	69
Yht.	284

Taulukko 4.4 *Tehtävästä 3 poimitut opiskelijoiden väärät vastaukset kategorisoituna*

esimerkiksi laskuvirheen tyyppiä ja syytä. Tehtävän 1 laskuvirheitä pystyttäisiin jakamaan esimerkiksi virheisiin itseisarvon, lukujen negaation tai neliöjuuren käytössä. Likiarvon antaminen vastaukseksi itseisarvotehtävässä ilmaisee, että opiskelija on ratkaissut tehtävän syöttämällä sen (yleensä suoraan) laskimeen, mikä ei yleisessä tapauksessa ole hyväksyttyä. Tarkastelemalla näiden opiskelijoiden koko vastausta välivaiheineen ilmenee useasti, että opiskelija on laskenut ja kirjannut hyväksyttävän vastauksen ennen likiarvon laskemistaan. Tarkastelemalla vain lopullista vastausta ei näitä tilanteita huomata, ja aikaisemmat vastaukset tulee ottaa huomioon tarkastusalgoritmia kehittäessä

Yksi virhetyyppi on myös kirjoitusvirhe. Mikäli MathCheckiä käyttävälle verkkosivulle ei ole toteutettu matematiikan ladontaa, opiskelijan ainoa kosketus kirjoitukseensa on ohjelmointikoodimainen teksti. Varsinkin ohjelmointia harrastamattomalle virheiden huomaaminen tästä voi olla erittäin vaikeaa aikarajojen sisällä, ja pienikin virhe muuttaa koko vastauksen. Tähän löytyy tehtävästä 1 esimerkki vastauksessa

$$\frac{5^1}{2} - 3,$$

joka on kirjoitettu MathCheckille "frac(5 ^(1))(2) - 3", vaikka opiskelija on selvästi tarkoittanut vastata

$$5^{\frac{1}{2}} - 3,$$

kirjoitettuna "5 ^(frac(1)(2)) - 3". Tässä analyysissä vastaus kuuluu laskuvirhekategoriaan, ja kuuluisi sinne vaikka vastaus olisikin kirjoitettu oikein.

Tämänlaisten kirjoitusvirheiden tunnistaminen koneellisesti on haastavaa. Jotta tietokone voisi edes epäillä virheen olevan vain kirjoitusvirhe, sen pitäisi käydä läpi lähes kaikki mahdolliset permutaatiot vastauksen merkkien sijaintien ja erehdyksessä kirjoitettavien merkkien suhteen. Vertailuun täytyisi ottaa huomioon myös kuinka monen merkin ero on pelkkä kirjoitusvirhe eikä oleellinen virhe. Vaikka kone epäilisi virheen olevan kirjoitusvirhe, voi se siltikin olla jokin muu virhe. Kirjoitusvirheiden ongelmallisuus pienenee, kun otetaan myös perustelut huomioon. Moni kirjoitusvirhe (ja jotkut muutkin virheet) voidaan hyväksyä, jos niitä edeltävät perustelut ovat kattavia ja oikeellisia.

4.4 Perustelujen tarkastus

Matemaattisten vastausten perustelut ovat joko tekstiä, matematiikan kaavoja tai useammin näiden yhdistelmiä; perusteluille ei ole yleensä muotovaatimuksia. Ihmistarkastajan on suhteellisen helppo tarkastaa perustelujen oikeellisuus ja riittävyys,

toisin kuin tietokoneen. Mikäli perustelu koostuisi pelkästä loogisesta ketjusta tehtävänannosta vastaukseen, ohjelmat kuten MathCheck 4.1.2 pystyvät kyllä tarkastamaan sen oikeellisuuden. Hyvä ladonta-algoritmi voi poimia perusteluista matemaattiset ketjut, mutta mahdollinen jäljelle jäävä teksti täytyy myös tarkastaa.

Tähän tarkastukseen voi olla mahdollista käyttää samoja menetelmiä kuin esseiden tarkastamiseen. Luvussa 3 esitetyistä menetelmistä luonnollisen kielen käsittely ei ole tarkoitukseen sopiva, sillä matemaattisen vastauksen perustelu täytyy olla ymmärrettävä, mutta ei välttämättä kaikkien kielensääntöjen mukainen. Muut esitetyt menetelmät voivat soveltua yhtä hyvin matemaattisten perustelujen tarkastamiseen.

Tarkastusmenetelmien testaamiseen voidaan käyttää Matlabin tarjoamia hermoverkostojen ja LSA -menetelmän kirjastoja. Insinöörimatematiikka A3 -kurssille järjestetystä sähköisestä tentistä voidaan ottaa yhden kysymyksen valmiiksi arvioidut vastaukset automaattitarkastusmenetelmien testaamiseen. Tentissä oppilas sai vastattavakseen yhden yhdeksästä mahdollisesta kysymyksestä, joka sisälsi kolme kohtaa a, b ja c. Kohta c oli kysymyksen muuttujien lukuarvoja lukuun ottamatta sama jokaisessa kysymyksessä, ja muut kohdat erosivat myös tehtävänannoltaan. Kohdan c tehtävänanto on kuvassa 4.2. Kaikkiaan tenttivastauksia oli 374, jotka jakautuivat tasaisesti yhdeksään eri kysymykseen.

Yhden kysymyksen vastauksia oli siis keskimäärin 42, joka on liian pieni joukko hyvään analyysiin. Sen sijaan ottamalla jokaisesta kysymyksestä vain kohta c analysoitavaksi, joukon koko on 374, joka on riittävän iso. Analyysissä täytyy vain huomioida, että lukuarvot vastauksen oikeellisuuden ja kysymyksen mukaisesti. Vastausten analyysi voidaan kuitenkin olettaa tehdyksi osan 4.3 mukaan, ja mikäli tarkastusmenetelmä ei löydä korrelaatiota lukuarvojen ja vastauksen saamien pisteiden välillä, niiden ei pitäisikään vaikuttaa arvosteluun. Yhdestä tehtävän kohdasta sai korkeintaan 2 pistettä, eli jokaiselle vastaukselle oli annettu arvosanaksi joko 0, 1 tai 2. Vastauksien arvosanajakauma oli hyvin tasainen.

c) Olkoon $p(t)$ populaation koko, missä t on aika vuosina. Populaation kokoa voidaan mallintaa differentiaaliyhtälöllä $p'(t) = r\,p(t)\left(1 - \frac{p(t)}{k}\right)$, missä positiivinen vakio r merkitsee kasvuvauhtia ja positiivinen vakio k on ympäristön kantokyky. Tässä tehtävässä $r = \frac{1}{3}$ ja $k = 200$. Alussa populaation koko $p(0) = 100$. Ratkaise populaation koko $p(t)$, piirrä populaation kokoa kuvaava kuvaaja ensimmäisen kymmenen vuoden osalta (vihje: plot) ja ratkaise koska populaation määrä on kasvanut 40%. Piirrä tämä piste samaan kuvaajaan (vihje: hold on; plot(aika,arvo,'r.')).

Kuva 4.2 *Kohdan c tehtävänanto*

Ratkaisu kohtaan c)

Kirjoita tähän perustelut ja laskut

```
syms p(t) r k t y

yht = diff(p,t) == r*p*(1-(p/k))
ehto = p(0) == 100

p(t) = dsolve(yht,ehto)
simplify(p(t))
r = 1/3;
k = 200;
x = [1:10];
y = (100*k*exp(r*x))/k+100*exp(r*x)-100;
plot(x,y)
hold on

solve(140==((100*k*exp(r*t))/k+100*exp(r*t)-100),t)
aika = double(ans);
arvo = (100*k*exp(r*aika))/k+100*exp(r*aika)-100
plot(aika,arvo,'r.')
```

Vastaus kohtaan c)

Populaation kokoa kuvaaja funktio on

p(t) = (100*k*exp(r*x))/k+100*exp(r*x)-100

Kysytyn ajan likiarvo on

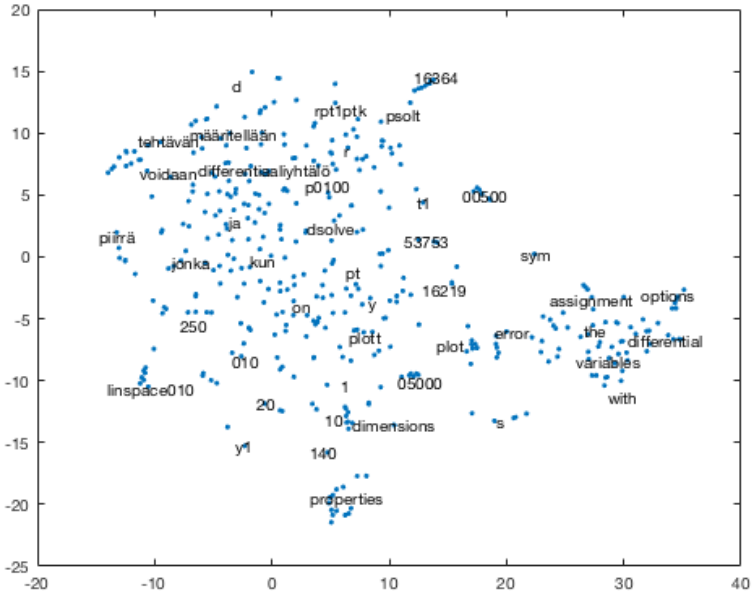
0.55 vuotta

Kuva 4.3 *Yksi vastaus kohtaan c*

Tenttiin vastattiin Matlab ohjelmalla, ja vastaukset palautettiin PDF muodossa. PDF-tiedostoista saatiin suurin osa tekstistä poimittua tavalliseksi koneluettavaksi tekstiksi, mutta kaikki funktiot ja muuttujat jotka oppilas oli kirjoittanut Matlabin omalla syntaksilla tallentuvat PDF-tiedostoon kuviksi, joita yksinkertaiset tekstinpoimintasovellukset eivät pysty muuttamaan koneluettaviksi. Kurssin ja tentin luonteen vuoksi oppilaat saivat käyttää hyväksi MatLabin funktioita ja menetelmiä, ja monen perustelu olikin lähinnä pelkkää MatLabin koodia. Teksti/kuva ongelmia ei siis ollut paljoa, mutta analyyseihin vaikuttaakin nyt enemmän se, että teksti on koodikieltä, jossa välimerkkejä ei tarvita. Nyt siis 'sana'-määritelmä ei ole yksiselitteinen, toisin kuin tekstin analysointi yleensä olettaa. Kuvassa 4.3 on yhden opiskelijan antama vastaus.

4.4.1 Hermoverkosto

Matlab tarjoaa hermoverkostoihin käytettäväksi konvoluutio- (CNN tai ConvNet) ja LSTM-hermoverkkoja [43]. Tähän analyysiin käytettiin LSTM-verkkoa yhdistettynä luokittelu tasoon. LSTM-verkko sisältää hermoja, jotka "muistavat"aikaisempien



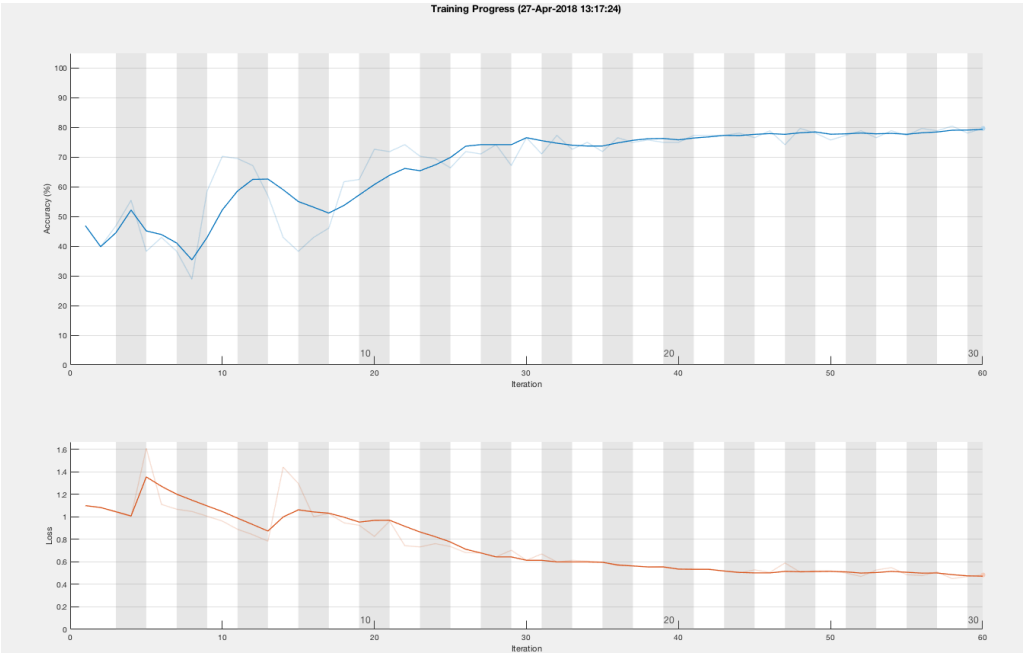
Kuva 4.4 *Sanojen vektorit koulutusavaruudessa*

verkon läpiajojen tuloksia. Näiden ansioista LSTM-verkko soveltuu erityisen hyvin häiriöisille sisääntuloille, ja se kykenee toimimaan kontekstista riippumattomien kielten kanssa [36].

Analyysiä varten vastaukset ja niille annetut arvosanat jaettiin koulutus- ja testausosiin. Testausosan koko oli noin 21 vastausta, ja se sisälsi yhtä monta jokaista saman arvosanan saanutta vastausta. Jaon lisäksi jokaisesta vastauksesta poistettiin välimerkit ja pienennettiin isot kirjaimet. Tämän jälkeen vastaukset muutettiin taulukoiksi, joissa jokainen solu koostui yhdestä sanasta. Koulutusosan vastaukset vektoroitiin ja istutettiin semanttiseen vektoriavaruuteen (vrt. LSA).

Kuvassa 4.4 on esitettynä yhdestä ajokerrasta poimitut sanavektorit projisoituna kaksiulotteiseen näkymään. Kuvasta nähdään kuinka moni sana on vain numero tai kirjain, koska kuten aiemmin mainittiin, moni vastaus sisälsi lähinnä matemaattista MatLab koodia. Tarkempaa tulosta varten voi olla hyödyllistä poistaa joitakin sanoja, esimerkiksi yksittäiset kirjaimet jotka ovat joko ennalta määritetyt tai opiskelijoiden vapaasti valittavissa olevia muuttujia. Tämä ei tietenkään poista kaikkia vapaita muuttujia, sillä niille voi antaa pitempiäkin nimiä. Numeroiden poisto voi tuottaa tarkempia tuloksia, mutta ne voivat myös olla perusteluille tärkeitä.

Avaruuteen istutetut sanat syötettiin hermoverkon kouluttajalle (Kuva 4.5). Kuvassa ylempi käyrä näyttää verkoston saavuttaman tarkkuuden koulutusajojen suhteen, ja alempi käyrä ilmoittaa hermoverkon tietohukan koulutusajojen suhteen. Koulu-



Kuva 4.5 *Matlabin neuraaliverkon koulutus*

tusajoja tehtiin jokaiselle koulutusosalle yhteensä 60. Annetuilla koulutusosilla hermoverkoston tarkkuus oli keskimäärin 80%, ja tietohukka 0,5.

Koulutuksien jälkeen verkolle annettiin testausosan vastaukset arvioitavaksi, ja verkon antamia arvosanoja verrattiin valmiisiin arvosanoihin. Analyysialgoritmi ajettiin kuusi kertaa, jokaisella kerralla eri koulutus/testus jaolla. Jokaisen ajokerran jako vaikutti sanojen vektorointiin, ja siten myös verkon koulutukseen. Verkko antoi keskimäärin 68,5% tarkkuuden antaa testijoukolle saman arvosanan kuin ihminen oli antanut. Syytä koulutuksen ja testauksen tarkkuuden erolle ei tällä hetkellä ole tiedossa. Testauskoodi löytyy liitteestä B.

Saavutettu tarkkuus ei ole erityisen hyvä, mutta suhteessa testidatan ongelmiin, se on riittävä. Jokaisen ajokerran tarkkuus erosi huomattavasti toisistaan. Tämä kertoo, että joko menetelmä ei sovellu tarkoitukseen, tai sille syötetyt tekstit eivät ole menetelmälle sopivia. Parempiin tuloksiin voidaan päästä esityöstämällä vastauksien sanoja, ja tutkimalla MatLbin hermoverkosto-ominaisuuksien asetuksia tarkemmin.

4.4.2 Piilevä semantiikka

LSA-menetelmän testaus alkoi samoin kuin hermoverkoston testaaminen: Vastaukset jaettiin osiin, ja istutettiin semanttiseen avaruuteen vektoreina. Istutetuista koulutusosan vektoreista luotiin LSA-avaruus, johon sitten testiosan istutetut vektorit sovitettiin. LSA-avaruuden koulutusvektoreista haettiin testausvektorien k lähintä

naapuria. Testivektoreiden arvosanaksi valittiin löydettyjen koulutusvektorien valmiista arvosanoista laskettu keskiarvo.

Esitetty menetelmä oli niin paljon nopeampi ja kevyempi suorittaa kuin hermoverkoston kouluttaminen, että sille voitiin tehdä optimointia tiettyjen muuttujien suhteen. LSA-avaruuden ulottuvuuksien määrä valittiin väliltä $[5, 80]$, ja k lähimmän naapurin k väliltä $[1, 9]$. Algoritmi voitiin näillä muuttujilla myös suorittaa useampia kertoja lyhyemmässä ajassa kuin yhden hermoverkoston kouluttaminen. Testauskoodi löytyy liitteestä C.

Jokaisella ajokerralla saatiin eri optimit LSA-avaruuden ulottuvuuksille ja k lähimmän naapurin k arvoille. Tästä voisi päätellä, että näiden arvojen valinnalla ei ole suurta merkitystä lopputulokseen. Samoin kuin hermoverkoston testauksessa, tarkkuus vaihteli suuresti ajokertojen välillä ja keskimääräinen tarkkuus oli noin 67%. LSA antoi useammin paremman arvosanan kuin valmiiksi annettu. Noin 10% vastauksista sai 2 pistettä suuremman arvosanan, mutta yksikään vastaus ei saanut 2 pistettä pienempää arvosanaa.

5. OHJELMALLINEN TARKISTUS: PUNAKYNÄ

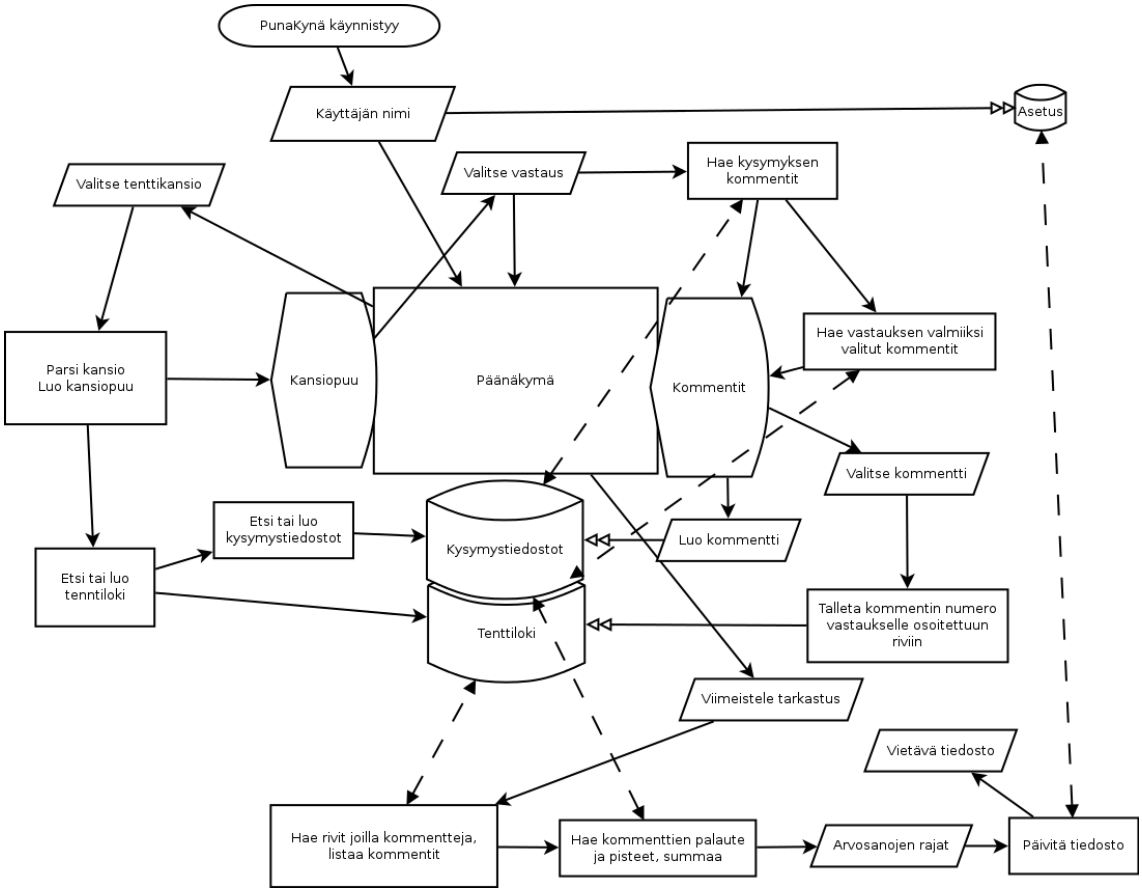
Tämän työn ohessa on tehty PunaKynä niminen ohjelma, jonka tarkoituksena on tuoda automaattista tenttien tarkastamista myös matematiikan tenteille. Ohjelman ensisijainen käyttötarkoitus on helpottaa tentinhallintajärjestelmästä saatujen vastaustiedostojen käsittelyä, ja nopeuttaa vastauksien arvostelua. Tämän lisäksi ohjelman toimintaa on tarkoitus laajentaa automaattisen tarkastamisen suuntaan.

5.1 Ohjelman toiminta

PunaKynän käyttö alkaa valitsemalla kansio, jossa tarkistettavat vastaukset ovat. Kuva 5.1 esittelee ohjelman toimintaa käyttäjän valinnoilla, jotka lähtevät kaaviossa päänäkymästä tai siihen liitetyistä näkymistä. Kansion valintaa vastaa kaavion kohta "Valitse tenttikansio". Kansiorakenteesta saadaan tentin ja kysymysten tunnisteet, joille luodaan tiedostot, sekä opiskelijoiden vastaukset, joita voi sitten selata kansioselaimella. Tässä vaiheessa ohjelma voi jo automaattisesti arvioida tyhjät vastaukset, joille EXAM-järjestelmä luo opiskelijalle kansion josta puuttuu liitetiedosto. Samoin voidaan tunnistaa virheelliset palautukset, jos esimerkiksi tiedosto on väärää muotoa tai sen koko on nolla tavua, ja antaa kyseisille palautuksille virheeseen sopiva palaute automaattisesti.

Kysymyksille ja tentille luodut tiedostot ovat csv-tiedostoja, joihin tallennetaan käyttäjän tekemät arvostelut. Kysymystiedostot sisältävät kysymykselle luodut kommentit, joihin tenttitiedostoissa viitataan niiden rivinumeroilla. Kommenttirivi koostuu sanallisesta palautteesta, kommentin pistearvosta ja mahdollisesta automaattitarkastuskoodista. Automaattitarkastuskoodi on nykyisessä versiossa perl-kielen säännöllisen lausekkeen tyylinen avainsana lista, johon on lisätty mahdollisuus viitata tarkastettavan tiedoston nimeen ja useamman avainsanan etsiminen järjestyksessä ja ilman järjestystä.

Valitessa ohjelmassa uuden opiskelijan vastauksen (Kuvan 5.1 kaaviossa kohta "Valitse vastaus"), ohjelma näyttää vastaustiedoston pääikkunassa, lukee vastausta vastaavasta kysymystiedostosta sen kommenttirivit ja tarjoaa ne tarkastajalle. Tarkastaja voi näistä kommenteista valita opiskelijan vastaukseen sopivat kommentit



Kuva 5.1 *Yksinkertainen PunaKynän käyttökaavio*

(Kuvan 5.1 kaaviossa kohta "Valitse kommentti"), luoda uusia kommentteja, tai kirjoittaa henkilökohtaisen kommentin. Tentin voi myös tarkastaa automaatiolla, jolloin opiskelijan vastauksista poimitaan kaikki mahdollinen teksti, johon jokaisen kommentin automaatiotarkastuskoodia verrataan. Jos koodi täsmää, ohjelma valitsee kyseisen kommentin käyttäjän puolesta. Kommenttien tarkastusta voi myös kontrolloida siten, että tietyt kommentit tarkastetaan aikaisempien kommenttien tarkastuksen tulosten mukaan.

Valitut kommentit ja henkilökohtainen kommentti tallennetaan tenttitiedostoon, jossa pystyrivit vastaavat kysymyksien tunnistetta, ja vaakarivit opiskelijoiden tunnistetta. Tässä vaiheessa tiedostoon tallentuu kommenttiriveistä ainoastaan niiden tunnisteet. Näin tentin arvostelija voi arvostelun aikana ja ennen viimeistelyä vapaasti muuttaa kommenttirivien sisältöä.

Kun tentin tarkastus on valmis, se viedään ulkoiseen tiedostoon (Kaaviossa 5.1 kohta "Viimeistele tarkastus"), jonka asettelu riippuu alustasta, johon tarkastus palautetaan. Tässä vaiheessa ohjelma yhdistää jokaiselle opiskelijan vastaukselle valitut kommenttien rivinumeron niiden kommenttitekstiin, liimaa jokaisen tekstin yhteen

ja laskee kommenttien pisteiden summan. Jos tentissä on useampi kuin yksi kysymys, tämä toistetaan jokaiselle eri vastaukselle. Kun jokaisen opiskelijan pisteet on kerätty, ohjelma kysyy käyttäjältä arvosanojen alarajat, joiden perusteella se täyttää palautustiedostoon opiskelijoiden saamat kommenttitekstit, kokonaispisteet ja näillä saavutetun arvosanan. Yhteen liimatut kommenttitekstit ja pisteet korvaavat tässä vaiheessa tenttitiedostossa kommenttien rivinumerot, jotta mahdolliset muutokset kommenteissa eivät muuta jo arvosteltujen vastausten arvoa.

5.2 Ohjelman tulevaisuus

Diplomityön laajuuteen nähden PunaKynä jää täydestä potentiaalistaan vajaaksi. Sitä kuitenkin voidaan jatkokehittää, ja on toivottavaa että näin tehdään.

5.2.1 Avoin lähdekoodi

Punakynä on tehty käyttäen LGPLv3-lisensöityjä [14] QT-kirjastoja [29], jotka antavat oikeuden lisensoida niitä käyttävä ohjelma vapaasti. Kun PunaKynän toiminta on varmistettu täysin toimivaksi, ja ohjelmakoodi on sievennetty ja hyvin kommentoitu, se on tarkoitus antaa vapaaseen levitykseen lähteineen. Näin sen kehitys voi jatkua ja haaraantua vapaasti täyttämään erilaisia tarpeita eri ympäristöissä, ja kehitys/ylläpito vastuu voidaan vapauttaa. [23]

QT-kirjastojen lisäksi PunaKynä käyttää Valmarin MathCheck-ohjelman [46] komentorivikäännöstä ja Poppler-kirjastoilla [12] luotua minimaalista PdfToTxt-ohjelmaa. Molemmat ovat PunaKynälle kolmannen osapuolen ohjelmia, ja ne voidaan tarvittaessa vaihtaa toisiin ohjelmiin. MathCheckin korvaava ohjelmisto tulee ottaa vastaan ja palauttaa standardivirroista tarkastettavan tekstin UTF-8 formaatissa, ja PdfToTxt-ohjelman korvaavan ohjelman tulee ottaa komentorivikutsussansa parametrina käännettävän tiedoston sijainnin, ja tulostettava saatu teksti standardi virtaan. Poppler on lisensöity GPL [13] lisenssillä, jonka ehtojen mukaan PdfToText-ohjelman mukana täytyy antaa sen lähdekoodi, ja Poppler-kirjastoja käyttävät ohjelmat täytyy myös lisensoida GPL lisenssillä. Tämän takia se ei ole integroituna PunaKynän koodiin; PunaKynä, ja sen johdannaisohjelmat ovat vapaasti lisensoitavissa.

5.2.2 Arvostelut useammasta lähteestä

PunaKynä kysyy käynnistyessään käyttäjän nimeä. Se tulostuu tentti- ja kysymystiedostojen tekstikenttien perään, joista voidaan tarkistaa, kuka tarkastuksen on

tehnyt. Jos voitaisiin olettaa, että jokainen arvostelija käyttäisi samoja kommentteja (eli samaa kysymystiedostoa) ja arvostelisi vain ennalta määrätyt vastaukset, ei arvostelujen yhdistäminen tuottaisi ongelmia. Monipuolisten käyttömahdollisuuksien takaamiseksi ohjelman on oltava valmis yhdistämään ristiriitaisetkin tiedostot.

Kysymystiedostoja yhdistäessä pitää ottaa huomioon muutetut, poistetut, lisätyt ja siirretyt rivit. Siirretyt ja lisätyt rivit voidaan löytää ja yhdenmukaistaa muokkaamalla toisen lähteen tenttitiedoston kommenttirivien numerot vastaaviksi. Muutetut ja poistetut rivit eivät löydy yhtä helposti, joten käyttäjän vastuulle voidaan jättää näiden tarkistaminen manuaalisesti, ja tämän jälkeen muokata tenttitiedostoa kuten aiemmin. Käyttäjän valitessa miten lähteet yhdistetään, ja miten tenttitiedostossa ilmenevien ristiriitojen kanssa toimitaan, voidaan näyttää, kuka on tehnyt mitkäkin muutokset ja arvostelut, ja tämän avulla päätetään, kenen työt korvataan toisella.

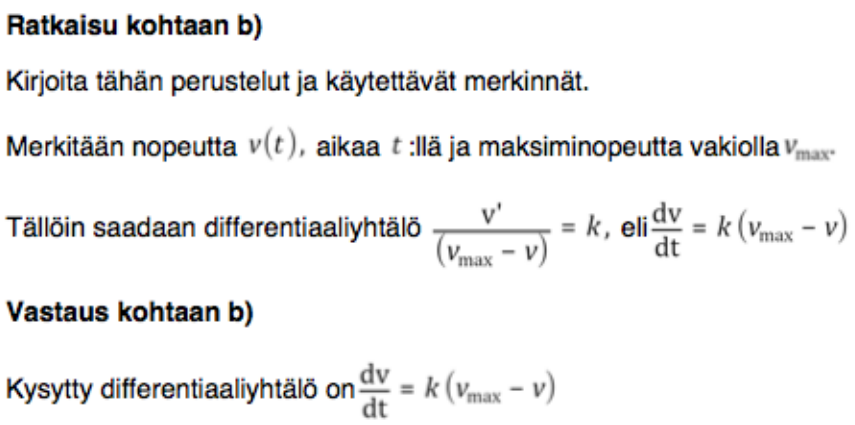
5.2.3 Matematiikan lukeminen

PunaKynän automatisoinnin suurin kynnys on tällä hetkellä matematiikan lukeminen opiskelijoiden vastauksista. MLE sallii käyttää LaTeX-syntaksin matematiikkaa, joka näkyy editorissa ja ulostulon PDF-tiedostossa LaTeX ladonnalla, ja sen saa PDF-tiedostosta ulos puhtaana LaTeX-koodina/tekstinä. Alkuvaiheen opiskelijoilta (ennen kandidaatin työtä) ei voida kuitenkaan vaatia ilman perusteellista perehdytystä LaTeX-koodin kirjoitustaitoja, joten on oletettava opiskelijoiden kirjoittavan vastauksiansa puhtaana tekstinä, tai useimmin käyttäen Live Editorin valmiita yhtälöpohjia. Valmiit yhtälöpohjat näkyvät ulkoasultaan täysin samalta kuin LaTeX-koodina kirjoitetut yhtälöt, mutta tuotettavaan PDF-tiedostoon ne tallentuvat kuvina. Ilman kunnollisia ohjelmia, näistä kuvista ei saa automaattisesti ulos tarvittavaa tekstiä, jonka avulla vastausta voisi arvioida (vrt. kuvat 5.2 ja 5.3).

Jotta edelleen kehitetyn PunaKynän algoritmeista saataisiin kaikki hyöty, täytyy löytää joko parempi matematiikan kirjoittaja, joka tuottaa koneen tunnistettavaa matemaattista tekstiä, tehokas OCR-ohjelma [35], joka tunnistaa ja tuottaa matemaattista tekstiä, tai toivoa että MATLAB kehittää Live Editoriansa tuottamaan puhdasta tekstiä myös valmiita yhtälöitä käyttäessä.

5.2.4 Automaatio

Olettaen, että opiskelijan vastauksesta saadaan kaikki teksti koneellisesti luettua, voidaan PunaKynän automaattista tarkastusta jatkokehittää. Tämä toteutuu jo



Kuva 5.2 *Opiskelijan vastaus PDF-lukijassa*

```
Ratkaisu kohtaan b)
Kirjoita t{\"a}h{\"a}n perustelut ja k{\"a}ytett{\"a}v{\"a}t merkinn{\"a}t.
Merkit{\"a}{\"a}n nopeutta          aikaa   :ll{\"a} ja maksiminopeutta vakiolla
T{\"a}ll{\"o}in saadaan differentiaaliyht{\"a}l{\"o}                  eli
Vastaus kohtaan b)
Kysytty differentiaaliyht{\"a}l{\"o} on
```

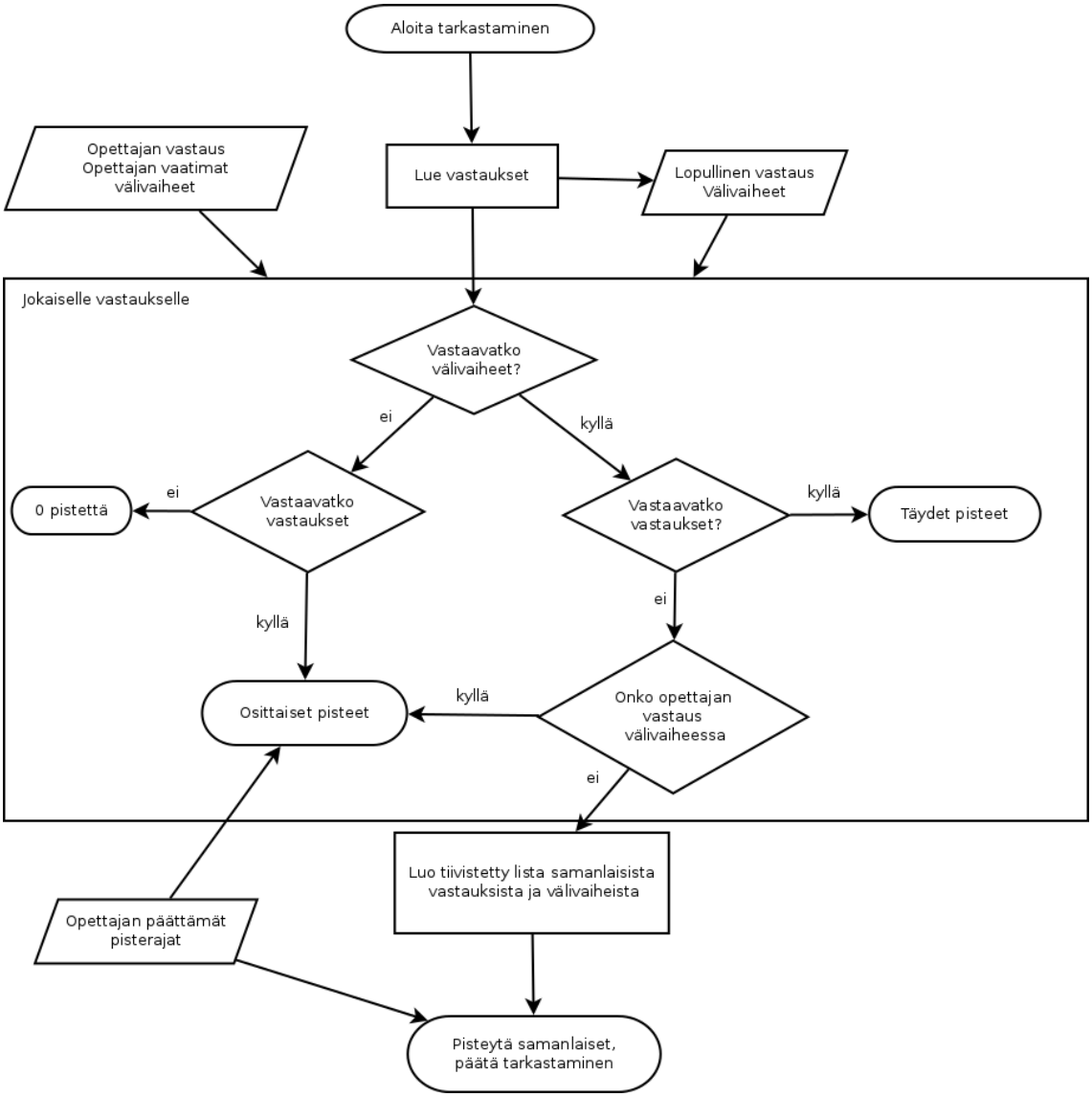
Kuva 5.3 *Opiskelijan vastaus tekstinä*

varsinkin silloin, kun tentti on tehty MathCheck ohjelmalla, ja palautettu HTML-tekstinä. Automaattiselle tarkastukselle voidaan luoda esimerkiksi kuvan 5.4 mukainen algoritmi.

Kun automaattinen tarkastaja on valmis, voidaan sitä jatkokehittää oppivammaksi. Algoritmi voi tallettaa useamman tentin tarkastuksesta samanlaisuuksia, ja luoda näistä oman tarkastustietokannan. Jossain vaiheessa ohjelma voi alkaa ehdottaa tarkastajalle aikaisempien tarkastusten mukaan mallinnettuja arvosteluja, joita hyväksymällä tai hylkäämällä tietokanta kehittyy entistä tarkemmaksi. Kun ohjelma on saanut tarpeeksi hyväksyttyjä ennakkoarvosteluja, se voi antaa arvosanat käyttäjältä kysymättä, jolloin ohjelma on saavuttanut täyden automatisaation, eikä tarkastajaa tarvitse häiritä kuin uusien kysymystyyppien ilmaantuessa.

5.3 Käyttöohje

Käynnistettäessä PunaKynää ensimmäistä kertaa, se kysyy käyttäjän nimeä, ja seuraavilla käynnistys kerroilla vain varmistaa nimen. Jos käyttäjä on vaihtunut edelliseltä käynnistys kerralta, PunaKynä tarjoaa listan aiemmin syötetyistä nimistä ja kentän mihin voi syöttää uuden nimen. Nimen voi myös jättää tyhjäksi.

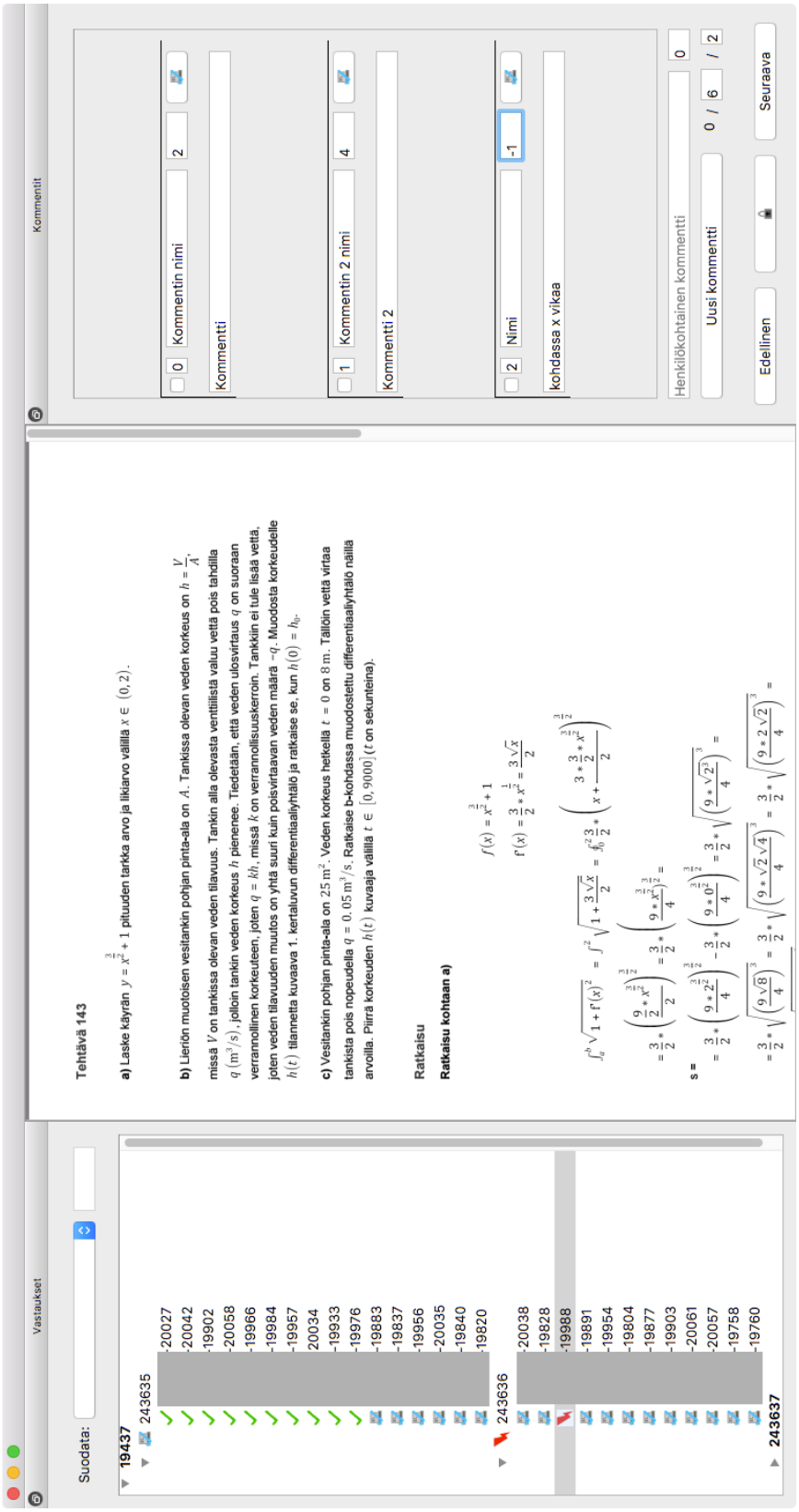


Kuva 5.4 *Automaattisen tarkastajan algoritmi*

Pääkäyttöliittymä koostuu kolmesta osasta: kansioselain (vasemmalla), kommenttiikkuna (oikealla) ja vastausikkuna (keskellä). Tämä on esitetty kuvassa 5.5. Kansioselaimen ja kommentti-ikkunan voi irrottaa pääikkunan reunoista ja liikuttaa vapaasti vetämällä niiden otsikko palkkeja hiiren osoittimella.

Tentintarkastusoperaatio aloitetaan valitsemalla työkalupalkista 'Tentti->Avaa uusi tentti', ja valitsemalla kansio, jossa kysymykset ovat alikansioina. Tämän jälkeen PunaKynä kysyy, mistä tenttikansio on haettu, ja missä tiedostoformaatissa vastausten pitää olla. Mikäli PunaKynällä on jo aiemmin avattu tarkastettava tentti, sen voi valita 'Tentti' valikon kohdasta 'Avaa vanha tentti' listasta. Lista sisältää korkeintaan kymmenen viimeksi avattua tenttiä.

Kysymyksille lisätään pisteytettyjä kommentteja painamalla 'Uusi kommentti' pai-



Kuva 5.5 *PunaKynän päänäkymä, josta opiskelijanumerot piilotetu*

niketta. Kysymyksillä voi olla korkeintaan 64 eri valittavaa kommenttia. Jokaisella kommentilla on valintaruutu, pikanäppäin, nimi, kommenttiteksti ja pisteet. Lisäksi kommenteille voi asettaa avainsanat automaattiselle tarkastukselle painamalla painiketta. Kysymyksille tehdyt kommentit ovat valittavissa jokaiselle vastaukselle, ja jokaiselle vastaukselle voi myös kirjoittaa henkilökohtaisen kommentin pisteineen.

Tarkastettava vastaus voidaan valita kansioselaimesta hiirellä tai ne voi käydä läpi järjestyksessä painamalla 'Seuraava' (pikanäppäimenä 'Rivinvaihto') ja 'Edellinen' painikkeita. Kun valittuna on vastaus, on samalla valittuna sen kysymys kommenttien lisäämisen suhteen. Kansioselaimessa vastaukset näkyvät lihavoituina kun niitä ei ole katsottu, ja normaalina kun ne on. Jos vastaukselle on annettu kommentteja, sen eteen tulostuu tästä kertova merkki.

Jokaisen kommentin oikealla laidalla on painike, josta voi asettaa kommentille avainsanalistan automaattitarkastusta varten. Automaattitarkastaja etsii tarkastaessaan opiskelijoiden vastauksista annettuja avainsanoja. Avainsanoille pätevät samat ominaisuudet kuin Perl-kielen säännöllisille lausekkeille, ja avainsanojen etsimisen järjestykseen voi vaikuttaa rivinvaihdoilla ja erikoismerkillä. Myös kommenttien tarkastustusjärjestykseen voi asettaa tarkemman kontrollin. Valitsemalla yläpalkin 'Automaattitarkastus' valikosta 'Muokkaa tarkastajan toimintaa', voi kommenttien tarkastuksen ehdollistaa: esim. tarkasta kommentti 3 jos kommentit 1 ja 2 toteutuvat. Samalla toiminnolla voi myös asettaa vastaukselle konflikti-tilan mm. niihin tapauksiin, joissa automaattitarkastaja ei löytänyt millekään kommentille annettuja avainsanoja. Tässä tilassa olevia vastauksia ei merkitä tarkastetuiksi ennen kuin käyttäjä on itse tarkastanut ne.

Kun vastauksille on annettu kommentteja, ne voidaan viedä tenttijärjestelmän haluamaan tiedostoon. Vientioperaatioon kuuluu arvosanajakauman päättäminen, jossa annetaan jokaisen arvosanan alaraja. Tietyn arvosanan rajaa ei voi asettaa pienemmäksi kuin edellisen arvosanan raja on, eikä suuremmaksi kuin seuraavan arvosanan raja on. Korkeimman arvosanan rajan voi asettaa yli tentistä saatavien maksimipisteiden. Jos opiskelijalle on annettu kommenteilla enemmän pisteitä tehtävästä kuin sen maksimipisteet ovat, ne leikataan maksimipisteeseen. Jos arvosanarajat annetaan, PunaKynä viimeistelee arvostelut, täydentää järjestelmään vietävän tiedoston ja luo Excel-tiedoston, johon jokaiselle riville tulee opiskelijan numero, saavutettu arvosana ja tehtäväkohtaiset pisteet. Kansioselaimessa jokaisen arvostellun vastauksen kuvake muuttuu kansion kuvaksi osoittaakseen, että vastaus on arvosteltu ja arkistoitu.

Ensimmäistä kertaa onnistuneesti sulkeutuessaan PunaKynä luo suorituskansioonsa

Kurssi	Tentin tyyppi	Määrä
MAT-01160	Perustaitojen testi	508
MAT-01160	1. viikkokoe	231
MAT-01160	Tentti	143
MAT-01200	Matlabin alkeet	97
MAT-01210	Tentti	619
MAT-01230	Tentti	262
MAT-01260	Matlabin alkeet	1547
MAT-01260	1. viikkokoe	219
MAT-01260	Tentti	170
MAT-01310	Tentti	416
MAT-01330	Tentti	247
MAT-04600	3. välikoe	211
MAT-04600	2. välikoe	214
MAT-04600	Tentti	22
MAT-04600	1. välikoe	670
Yhteensä		5576

Taulukko 5.1 *PunaKynän käyttö lukuvuonna 2017-2018*

asetustiedoston. Tämä tiedosto, kuten myös valittuihin kansioihin luodut tentti- ja kysymystiedostot ovat ihmisluettavia tekstitiedostoja. Käyttäjän on mahdollista kirjoittaa ja muokata tiedostoja ohjelman ulkopuolellakin, mutta tällöin mahdolliset suoritusvirheet ovat käyttäjän itsensä vastuulla.

5.4 Käyttö

PunaKynän ensimmäinen alpha-versio oli valmis 10.5.2017. Tästä versiosta puuttui paljon ominaisuuksia, ja se toimi lähinnä todisteena konseptista. Kesän 2017 aikana sitä päivitettiin kovalla tahdilla, ja sitä käytettiin muutamiin tentteihin kokeilu ja testaus mielessä. PunaKynän ensimmäinen käyttökelpoinen versio, beta 1.7.1, oli käytettävissä 22.8.2017. Tämän jälkeen sen päivitystahti hidastui, ja päivitykset keskittyivät lähinnä virheiden korjaamisiin. Kirjoitushetkellä uusimman PunaKynän versionumero on 1.7.13.

PunaKynä oli käytössä monen eri matematiikan tentin tarkastamisessa seuraavana lukuvuonna. Taulukossa 5.1 on lueteltuna lukuvuoden 2017-2018 tentit, jotka tarkastettiin käyttäen PunaKynää. Tarkastettuja tenttejä oli yhteensä 5576 kahdeksassa eri kurssissa. MAT-01160-kurssin Perustaitojen testi tehtiin MathCheckillä, muissa tenteissä oli käytössä MLE. Jokainen tenttityyppi pitää sisällään mahdolliset uusintatentit.

Sähköiset tentit ovat vielä uusi käsite Tampereen teknillisessä yliopistossa, mutta

muutamista heikkouksista riippumatta niihin suhtaudutaan myönteisesti [20]. Nykyisessä EXAM-järjestelmässä PunaKynää voidaan käyttää vain kun tentti palautetaan erillisenä tiedostona. Jos tenttiin vastataan järjestelmän tarjoamaan vastauslaatikkoon, vastausta ei saada järjestelmästä ulos yhtä helposti. PunaKynän laajempi käyttöönotto vaatii siis joko järjestelmän muokkaamista arvostelun ulkoistamisen helpottamiseksi, tai kolmannen osapuolen ohjelman (esim. MLE) sisällyttämistä useamman kurssin vaatimuksiin.

Uusia ratkaisuja tenttien ja harjoitusten sähköistämiseen luodan jatkuvasti. Tällä hetkellä Tampereen teknillisen yliopiston matematiikan laboratoriossa yritetään luoda harjoitusjärjestelmää, joka pyörii TUT+ nimisessä järjestelmässä, tarkistaa opiskelijoiden vastaukset MathCheckillä, ja antaa vastauksille lopulliset palautteet PunaKynällä. Tenteissä käytettävä MLE on myös toivottavasti jossain vaiheessa korvattavissa vastaavalla, tai paremmin matematiikan kirjoittamista yhtäaikaisesti kone- ja ihmisluettavaksi tukevalla ohjelmalla, jos MLE itse ei päivity tähän soveltumammaksi.

6. YHTEENVETO

Esseiden automaattisen tarkastamisen historia alkoi yli 50 vuotta sitten. Viime vuosikymmenien aikana on saatu aikaiseksi useita erilaisia toimivia tarkastusohjelmia, joista muutamat ovat aktiivisessa käytössä eri instituutioissa. Monet ovat esittäneet vastahakoisuutta antaa koneille vastuuta arvostella luovaa kirjoitusta, jota vain luova ihminen voi ymmärtää. Vasta-argumenttina tähän puolestapuhujat vaivat näyttää, kuinka luovuutta ymmärtämätön tietokone arvioi tekstejä yhtä tarkasti kuin ihminen.

Vastaanpuhujien epäilyt eivät kuitenkaan ole täysin pohjattomia. Automaattitarkastajien kehittäjätkin myöntävät, ettei heidän ohjelmansa ole täydellisiä, varsinkaan luovuuden arvioinnissa. Tietokoneiden algoritmeja voidaan myös huijata, kun niiden toimintaperiaatteet tunnetaan [28]. Kaikki ohjelmat kuitenkin kehittyvät ajan myötä, ja automaattitarkastajat eivät ole poikkeus.

Suurin osa tenttien automaattitarkastamistyöstä on tehty esseetenttien tarpeisiin. Tämä ei kuitenkaan tarkoita, että automaattitarkastuksen toteuttaminen myös matematiikan tenteille ei olisi mahdollista. Tämä vaatii vain hieman erilaista otetta automatiikan toteuttamisessa. Matemaattiset vastaukset ovat yleensä lyhyempiä, mutta monipuolisempia. Niiltä myös odotetaan eri ominaisuuksia. Matemaattiselle kysymykselle on usein yksi tai ainakin rajoitettu määrä vastauksia. Esseen tarvitsee yleensä vain mainita joitain asioita aiheeseen liittyen, antamatta mitään konkreettista kaikenkattavaa lopullista vastausta.

Valmiita automaattitarkastusmalleja voidaan käyttää osana matematiikan tenttien tarkastamista, mutta ei sellaisenaan, eikä ainoana osana. Toisin kuin esseevastauksessa, matemaattisessa vastauksessa lopputulos on tärkeämpi, ja välivaiheet ovat rajoitetumpia. Esseen rakenne voi kulkea monia reittejä, aiheen ohi ja yli, mutta matematiikan vastauksen tulee kulkea tiettyjä raiteita, lähtien alkuoletuksista, kulkien loogisia seurauksia pitkin ja saapuen lopputulokseen. Matematiikan sanasto ja kielirakenne on myös laajempi ja vapaampi, eikä luonnollisen kielen välimerkkien säännöt päde matematiikan kielessä. Matemaattinen vastaus voi myös sisältää molempia kieliä. Ehkä matematiikan tenttien tarkastamista kannattaa lähestyä toisesta

näkökulmasta?

Vähin mitä voidaan tehdä, on ainakin helpottaa sähköisten järjestelmien hyödyntämistä. PunaKynä mahdollistaa kahdesta eri järjestelmästä saatavien tenttivastauksien arvioinnin järjestelmän tarjoamaa tapaa tehokkaammin. Sen oleellisimmat ominaisuudet ovat vastaustiedostojen hallinta, palautteenannon yksinkertaistaminen ja arvioinnin siirto takaisin tenttausjärjestelmään. Tällä hetkellä se tarjoaa myös yksinkertaisen sanahakuun perustuvan automaattisen arvioinnin, jota on tarkoitus kehittää edelleen kattavammaksi ja joustavammaksi. Jatkokehitys jäänee kuitenkin jonkun muun hoidettavaksi. Sivulta `https://github.com/LaaksonenV/PunaKyna` voi ladata Punankynän omaan kokeiluun. Samalle sivulle päivittyy myös sen lähdekoodi, kun se on jaettavassa muodossa.

LÄHTEET

[1] Y. Attali ja J. Burstein. “Automated Essay Scoring With e-rater v.2.0”. English. 2004-12. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2333-8504.2004.tb01972.x.

[2] B. P. Bergeron. “Optical mark recognition”. Tallying information from filled-in ’bubbles’. *Postgraduate Medicine* 104.2 (syyskuu 1998). URL: https://web.archive.org/web/20060613072246/http://postgradmed.com/issues/1998/08_98/dd_aug.htm (viitattu 07. 06. 2018).

[3] M. W. Berry, Z. Drma? Ja E. R. Jessup. “Matrices, Vector Spaces, and Information Retrieval”. *SIAM Review* 41.2 (1999), s. 335–362. ISSN: 00361445. URL: http://www.jstor.org/stable/2653077.

[4] A. Borsetta, C. Young, G. Lo ja K. Young. *FormScanner*. 2008. URL: http://www.formscanner.org/ (viitattu 07. 06. 2018).

[5] H. M. Breland, R. J. Jones, L. Jenkins, M. Paynter, J. Pollack ja F. Y. Fai. “THE COLLEGE BOARD VOCABULARY STUDY”. *ETS Research Report Series* 1994.1 (), s. i–51. DOI: 10.1002/j.2333-8504.1994.tb01599.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2333-8504.1994.tb01599.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2333-8504.1994.tb01599.x.

[6] J. Burstein, M. Chodorow ja C. Leacock. “CriterionSM Online Essay Evaluation: An Application for Automated Evaluation of Student Essays.” Teoksessa: 2003. URL: http://www.aaai.org/Papers/IAAI/2003/IAAI03-001.pdf (viitattu 15. 06. 2018).

[7] D. Callear, J. Jerrams-Smith ja S. Victor. “Bridging gaps in computerised assessment of texts”. Teoksessa: *Proceedings IEEE International Conference on Advanced Learning Technologies*. 2001, s. 139–140. DOI: 10.1109/ICALT.2001.943881.

[8] B. Cheang, A. Kurnia, A. Lim ja W.-C. Oon. “On automated grading of programming assignments in an academic institution”. *Computers & Education* 41.2 (2003), s. 121 –131. ISSN: 0360-1315. DOI: https://doi.org/10.1016/S0360-1315(03)00030-7. URL: http://www.sciencedirect.com/science/article/pii/S0360131503000307.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu ja P. Kuksa. "Natural Language Processing (Almost) from Scratch". *The Journal of Machine Learning Research* 12 (8. marraskuuta 2011), s. 2493–2537. ISSN: 1532-4435. URL: http://www.jmlr.org/papers/v12 (viitattu 12. 06. 2018).

[10] D. Dubin. "The most influential paper Gerard Salton never wrote". *Library Trends* 52.4 (22. maaliskuuta 2004), s. 748. URL: https://search.proquest.com/docview/220454097.

[11] S. T. Dumais. "Latent semantic analysis". *Annual Review of Information Science and Technology* 38.1 (tammikuu 2004), s. 188–230. ISSN: 1550-8382. DOI: 10.1002/aris.1440380105. URL: http:https://doi.org/10.1002/aris.1440380105.

[12] freedesktop.org. *Poppler*. URL: https://poppler.freedesktop.org/ (viitattu 24. 11. 2017).

[13] GNU. *General Public License*. URL: https://www.gnu.org/licenses/gpl-3.0.en.html (viitattu 24. 11. 2017).

[14] GNU. *Lesser General Public License*. URL: https://www.gnu.org/licenses/lgpl-3.0.en.html (viitattu 24. 11. 2017).

[15] J. Hattie ja H. Timperley. "The Power of Feedback". *Review of Educational Research* 77.1 (2007), s. 81–112. DOI: 10.3102/003465430298487. eprint: https://doi.org/10.3102/003465430298487. URL: https://doi.org/10.3102/003465430298487.

[16] M. A. Hearst. "The debate on automated essay grading". *IEEE Intelligent Systems and their Applications* 15.5 (syyskuu 2000), s. 22–37. ISSN: 1094-7167. DOI: 10.1109/5254.889104.

[17] J. Hirschberg ja C. D. Manning. "Advances in natural language processing". *Science* 349.6245 (2015), s. 261–266. ISSN: 0036-8075. DOI: 10.1126/science.aaa8685. eprint: http://science.sciencemag.org/content/349/6245/261.full.pdf. URL: http://science.sciencemag.org/content/349/6245/261.

[18] G. James. *Advanced Modern Engineering Mathematics*. 4. painos. Pearson, 2011.

[19] T. Kakkonen, N. Myller, E. Sutinen ja J. Timonen. "Comparison of Dimension Reduction Methods for Automated Essay Grading". English. *Journal of Educational Technology & Society* 11.3 (2008), 275–n/a. URL: https://search.proquest.com/docview/1287039684?accountid=27303.

[20] S. Koskinen. "Sähköinen arviointi matematiikan opetuksessa". Diplomityö. Tampereen Teknillinen Yliopisto, 2017.

[21] Z. F. Muhsen, A. Maaita, A. Odah ja A. Nsour. “Moodle and e-learning Tools”. English. *International Journal of Modern Education and Computer Science* 5.6 (heinäkuu 2013), s. 1–8. URL: https://search.proquest.com/docview/1627735504?accountid=27303.

[22] National Council of Teachers of English. *NCTE Position Statement on Machine Scoring. Machine Scoring Fails the Test.* URL: http://www2.ncte.org/statement/machine_scoring/ (viitattu 20. 11. 2017).

[23] Open Source Initiative. *Open Source.* URL: https://opensource.org/ (viitattu 24. 11. 2017).

[24] E. B. Page. “Computer grading of student prose, using modern concepts and software.” *Journal of Experimental Education* 62.2 (1994), s. 127. ISSN: 00220973. URL: http://search.ebscohost.com/login.aspx?direct=true&db=afh&AN=9612041568&site=ehost-live&scope=site.

[25] E. B. Page ja N. S. Petersen. “The computer moves into essay grading.” *Phi Delta Kappan* 76.7 (1995), s. 561. ISSN: 00317217. URL: http://search.ebscohost.com/login.aspx?direct=true&db=afh&AN=9503202277&site=ehost-live&scope=site.

[26] C. H. Papadimitriou, H. Tamaki, P. Raghavan ja S. Vempala. “Latent Semantic Indexing: A Probabilistic Analysis”. Teoksessa: *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. PODS ’98. Seattle, Washington, USA: ACM, 1998, s. 159–168. ISBN: 0-89791-996-3. DOI: 10.1145/275487.275505. URL: http://doi.acm.org/10.1145/275487.275505.

[27] A. Pollitt. “The method of Adaptive Comparative Judgement”. *Assessment in Education: Principles, Policy & Practice* 19.3 (2012), s. 281–300. DOI: 10.1080/0969594X.2012.665354.

[28] D. E. Powers, J. C. Burstein, M. Chodorow, M. E. Fowles ja K. Kukich. “Stumping e-rater:challenging the validity of automated essay scoring”. English. *Computers in Human Behavior* 18.2 (2002), s. 103–134. DOI: 10.1016/S0747-5632(01)00052-8. URL: https://www.sciencedirect.com/science/article/pii/S0747563201000528.

[29] QT Company. *QT.* URL: https://www.qt.io/what-is-qt/ (viitattu 24. 11. 2017).

[30] L. M. Rudner. *Bayesian Essay Test Scoring sYstem.* URL: http://echo.edres.org:8080/betsy/ (viitattu 07. 06. 2018).

[31] L. M. Rudner ja T. Liang. "Automated essay scoring using Bayes' theorem". *Journal of Technology, Learning, and Assessment* 1.2 (2002). URL: https://ejournals.bc.edu/ojs/index.php/jtla/issue/view/207.

[32] Y. Rui ja M. Ortega. "Information Retrieval Beyond the Text Document." *Library Trends* 48.2 (1999), s. 455. ISSN: 00242594. URL: http://search.ebscohost.com/login.aspx?direct=true&db=afh&AN=2713479&site=ehost-live&scope=site.

[33] G. Salton, A. Wong ja C. S. Yang. "A Vector Space Model for Automatic Indexing". *Commun. ACM* 18.11 (1975), s. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220. URL: http://doi.acm.org/10.1145/361219.361220.

[34] C. Sangwin. *About the STACK Project*. URL: https://github.com/maths/moodle-qtype_stack/blob/master/doc/en/About/index.md (viitattu 05. 03. 2018).

[35] H. F. Schantz. *History of OCR, Optical Character Recognition*. Recognition Technologies Users Association, 1982. ISBN: 0943072018.

[36] J. Schmidhuber. "Deep learning in neural networks: an overview". *Neural networks: the official journal of the International Neural Network Society* 61 (2015), s. 85–117. DOI: 10.1016/j.neunet.2014.09.003. URL: http://www.ncbi.nlm.nih.gov/pubmed/25462637.

[37] F. Sebastiani. "Machine Learning in Automated Text Categorization". *ACM Comput. Surv.* 34.1 (maaliskuu 2002), s. 1–47. ISSN: 0360-0300. DOI: 10.1145/505282.505283. URL: http://doi.acm.org/10.1145/505282.505283.

[38] E. T. Service. *ETS*. URL: https://www.ets.org/ (viitattu 15. 06. 2018).

[39] G. G. Shipra ja D. Rattan. "NATURAL LANGUAGE PROCESSING ITS TYPES". English. *International Journal of Advanced Research in Computer Science* 8.8 (syyskuu 2017). Copyright - Copyright International Journal of Advanced Research in Computer Science Sep 2017; Last updated - 2017-10-23. URL: https://search.proquest.com/docview/1953779933?accountid=27303.

[40] H. Sirkka, R. Pirkko ja S. Paula. *Tutki ja kirjoita*. Tammi, 2000.

[41] The Center for Teaching and Faculty Development, San Francisco State University. "Origins and Purposes of Multiple Choice Tests". URL: http://ctfd.sfsu.edu/teaching-practices/origins-and-purposes-of-multiple-choice-tests (viitattu 21. 10. 2017).

[42] The MathWorks. *Computer Algebra System*. URL: https://se.mathworks.com/discovery/computer-algebra-system.html (viitattu 26. 06. 2018).

[43] The MathWorks. *Deep Learning*. URL: https://se.mathworks.com/discovery/deep-learning.html#withmatlab (viitattu 26. 06. 2018).

[44] The MathWorks. *MATLAB Live Editor*. URL: https://se.mathworks.com/products/matlab/live-editor.html (viitattu 26. 06. 2018).

[45] S. Valenti, N. Francesca ja A. Cucchiarelli. "An overview of current research on automated essay grading". *Journal of Information Technology Education* 2 (2003), s. 313–330. URL: https://doi.org/10.28945/331.

[46] A. Valmari. *MathCheck*. URL: http://math.tut.fi/mathcheck/MathCheck2017_1.html (viitattu 05. 03. 2017).

[47] P. Wiemer-Hastings. "Latent semantic analysis". Teoksessa: *Proceedings of the 16th international joint conference on Artificial intelligence*. Citeseer. 2004, s. 1–14.

[48] R. Williams. "Automated essay grading: an evaluation of four conceptual models". Teoksessa: *New horizons in university teaching and learning: Responding to change*. Toim. M. Kulski ja A. Herrman. Perth Australia: Centre for Educational Advancement, Curtin University, 2001, s. 173–184. URL: http://hdl.handle.net/20.500.11937/11929.

[49] Ylioppilastutkintolautakunta. *Digitaalinen ylioppilastutkinto*. URL: https://www.ylioppilastutkinto.fi/ylioppilastutkinto/digitaalinen-ylioppilastutkinto (viitattu 26. 06. 2018).

[50] Ylioppilastutkintolautakunta. *Hyvän vastauksen piirteitä*. 26. maaliskuuta 2018. URL: https://drive.google.com/file/d/1aNoEGLaBxP_vfjIhJnEHTjejCvm-LHYh/view (viitattu 10. 09. 2018).

[51] Ylioppilastutkintolautakunta. *Matemaattisen tekstin kirjoittaminen*. URL: https://www.abitti.fi/fi/ohjeet/matemaattisen-tekstin-kirjoittaminen-abitissa/ (viitattu 26. 06. 2018).

A. INSINÖÖRIMATEMATIIKKA 123 -KURSSIN TENTIN VASTAUKSIA

Tehtävänanto	Laske $\lvert\sqrt{5}-3\rvert$		Laske $\lvert\sqrt{5}-3\rvert$		Laske $\lvert\sqrt{6}-4\rvert$	
	Vastaus	lkm.	Vastaus	lkm.	Vastaus	lkm.
Oikea vastaus	$3-\sqrt{5}$	80	$3-\sqrt{5}$	84	$4-\sqrt{6}$	79
Sieventämätön	$-\sqrt{5}+3$	4	$-\sqrt{5}+3$	3	$-\sqrt{6}+4$	5
	$-(\sqrt{5}-3)$	2			$-(\sqrt{6}-4)$	2
Väärä vastaus		9		6		7
	$\frac{7639}{10000}$	6	$\frac{7639}{10000}$	6	$\sqrt{6}+4$	5
	$\sqrt{5}+3$	5	$\sqrt{5}-3$	5	$\frac{31}{20}$	5
	$\sqrt{5}-3$	3	$\sqrt{5}+3$	3	$\lvert\sqrt{6}-4\rvert$	2
	$\lvert\sqrt{5}-3\rvert$	2	$\lvert\sqrt{5}-3\rvert$	3	10	2
	$(\frac{559}{250}-3)*(-1)$	1	$\frac{19}{25}$	2	$4+\sqrt{6}$	2
	$-(\sqrt{5}+3)$	1	$3+\sqrt{5}$	2	$\frac{3101}{2000}$	2
	1	1	$-\lvert\sqrt{5}-3\rvert$	1	$\frac{3101}{2000}$	2
	2.236068	1	4	1	$(\sqrt{6})^2-4^2$	1
	22	1	$\frac{3}{4}$	1	$-\frac{155051}{100000}$	1
	$25-3$	1	$\frac{5^1}{2}-3$	1	$4-\frac{1}{36}$	1
	$3+\sqrt{5}$	1	$\frac{1}{2}$	1	6^2-4	1
	4	1	$\frac{191}{250}$	1	$\frac{1}{36}-4$	1
	$\frac{19}{25}$	1	$\frac{763}{1000}$	1	$\frac{2}{3}$	1
	$\frac{191}{250}$	1	$\sqrt{4}$	1	$\frac{1550510257}{1000000000}$	1
	$\sqrt{5-3}$	1	$\sqrt{9-5}$	1	$\sqrt{6}-4$	1
	$\lvert-3\rvert$	1	$\lvert\frac{1}{2}\rvert$	1	$\sqrt{6}-\sqrt{2}$	1
	$\lvert-\frac{7639}{10000}\rvert$	1	$\lvert\frac{5^1}{2}-3\rvert$	1	x	1
	$\lvert\frac{22361}{10000}-3\rvert$	1	$\lvert\frac{7639}{10000}\rvert$	1	$\lvert 2\sqrt{3}-4\rvert$	1
	$\lvert\rvert$	1	$\lvert\sqrt{4}\rvert$	1	$\lvert\frac{4899}{2000}-4\rvert$	1
	$\lvert 5^{\frac{1}{2}}-3\rvert$	1	$\lvert\rvert$	1	$\lvert\sqrt{2*3}-4\rvert$	1
	0.763932	1	-4	1		
Yhteensä	25	128	24	129	24	126

Taulukko A.1 *Tehtävästä 1 poimitut opiskelijoiden vastaukset*

T.	Laske $\sin\frac{\pi}{4} - \sin\frac{7\cdot\pi}{4}$		Laske $\sin\frac{\pi}{4} - \sin\frac{3\cdot\pi}{4}$		Laske $\sin\frac{\pi}{4} - \sin\frac{7\cdot\pi}{4}$	
	Vastaus	l.	Vastaus	l.	Vastaus	l.
O.	$\sqrt{2}$	11	0	69	$\sqrt{2}$	16
S.	$\frac{2}{\sqrt{2}}$	1			$\frac{2}{\sqrt{2}}$	2
V.		27		13	0	24
	0	22	$\sin -\frac{\pi}{2}$	4		22
	1	13	$\sin\frac{-2\pi}{4}$	3	1	5
	-6	3	-1	2	$2\cdot\sin\frac{\pi}{4}$	3
	$-\sin\frac{6\cdot\pi}{4}$	3	$-\sin\frac{2\cdot\pi}{4}$	2	$\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}$	3
	$2\cdot\sin\frac{\pi}{4}$	3	$-\sin\frac{\pi}{2}$	2	$\frac{7071}{5000}$	3
	$\frac{7071}{5000}$	3	1	2	-1	2
	$\sin\frac{\pi-7\cdot\pi}{4}$	3	$\sin\frac{\pi-3\cdot\pi}{4}$	2	$-\sin 6\frac{\pi}{4}$	2
	$\sin\frac{\pi}{2}$	3	$\sin\frac{\pi}{2}$	2	$\frac{1}{2} - -\frac{1}{2}$	2
	$-\sin\frac{3\cdot\pi}{2}$	2	$\sin\pi$	2	$\sin\frac{\pi-7\cdot\pi}{4}$	2
	$-\sin 6\cdot\pi$	2	$\sin\frac{\pi}{4} - \frac{3\cdot\pi}{4}$	2	$\sin\frac{\pi}{2}$	2
	$\sin\frac{6\cdot\pi}{4}$	2	-180	1	$-\frac{6\pi}{4}$	1
	$\sin\frac{\pi}{4} - \sin\frac{\pi}{4}$	2	$-\frac{2}{\sqrt{2}}$	1	$-\frac{\sin 6\cdot\pi}{4}$	1
	-0.08209995	1	$-\frac{1}{2}\cdot\pi$	1	$-\frac{3}{2}$	1
	$-2\cdot\sin\frac{-3\cdot\pi}{4}$	1	$-\frac{5551}{5000}e - 16$	1	$-\sin 3\frac{\pi}{2}$	1
	$-\sin 6\cdot\frac{\pi}{4}$	1	$-\pi$	1	$-\sin\frac{6\cdot\pi}{4}$	1
	$-\sin\frac{3}{2}\cdot\pi$	1	$-\sin 2\frac{\pi}{4}$	1	$-\sin\frac{6\pi}{4}$	1
	$1 - -1$	1	$-\sin\frac{\pi}{4}$	1	$-\sin\frac{\pi+7\cdot\pi}{4}$	1
	$1 - 1$	1	$-\sin\frac{1}{2}\cdot\pi$	1	$-\sin\frac{\pi}{4}$	1
	$2\cdot\cos\pi\cdot\sin -2\cdot\frac{\pi}{3}$	1	$1 - -1$	1	$-\sin 6\cdot\frac{\pi}{4}$	1
	$2\cdot\cos\pi\cdot\sin -3\cdot\pi$	1	$2\cdot\cos\frac{\pi}{2} - \sin\frac{\pi}{4}$	1	$1 - -1$	1
	$3\cdot\frac{\pi}{2}$	1	$2\cdot\frac{\pi}{4}$	1	2	1
	$=$	1	2π	1	$2\cdot\sqrt{2}$	1
	$\frac{2}{\sqrt{2}} - -\frac{2}{\sqrt{2}}$	1	3	1	$45 - 7\cdot 45$	1
	$\frac{\sin 6\cdot\pi}{4}$	1	$=$	1	$\cos\frac{\pi}{2}\cdot\sin -3\frac{\pi}{4}$	1
	$\frac{\sqrt{2}}{2} - \frac{1}{\sqrt{2}}$	1	$\frac{1}{\sqrt{2}} - \frac{3}{\sqrt{2}}$	1	$\frac{-6\pi}{4}$	1
	$\frac{7017}{10000} -$	1	$\frac{2}{4}$	1	$\frac{1}{2}$	1
	$\frac{7071}{10000}$	1	$\frac{2}{\sqrt{2}}$	1	$\frac{1}{2} -$	1
	$\sin\frac{-3\pi}{2}$	1	$\frac{\pi}{4} - 3\cdot\frac{\pi}{4}$	1	$\frac{1}{2} - \frac{1}{2}$	1
	$\sin\frac{-6\cdot\pi}{4}$	1	$\frac{\pi}{4} - \frac{3\cdot\pi}{4}$	1	$\frac{2\cdot\sqrt{2}}{4}$	1
	$\sin\frac{-6\pi}{4}$	1	$\frac{\sqrt{2}}{2} - \frac{-\sqrt{2}}{2}$	1	$\frac{\pi}{2}$	1
	$\sin\frac{1}{\pi}$	1	$\sin\frac{-2\cdot\pi}{4}$	1	$\frac{19}{200}$	1
	$\sin\frac{3\pi}{2}$	1	$\sin -\frac{\pi}{4}$	1	$\frac{707}{1000} - -\frac{707}{1000}$	1
	$\sin\frac{\pi-7\pi}{4}$	1	$\sin\frac{\pi}{4} - 3\cdot\frac{\pi}{4}$	1	$\frac{707}{500}$	1
	$\sin\pi$	1	$\sin -2\cdot\frac{\pi}{4}$	1	π	1

	$\sin -\frac{6}{4} \cdot \pi$	1				$\sin \frac{-6 \cdot \pi}{4}$	1
	$\sin 3 \cdot \frac{\pi}{2}$	1				$\sin \frac{1}{7}$	1
	$\sin \frac{\pi}{4} - 7 \cdot \frac{\pi}{4}$	1				$\sin \frac{1}{\sqrt{2}} - \sin 7 \cdot \frac{1}{\sqrt{2}}$	1
	$\sin \frac{\pi}{4} - \frac{7 \cdot \pi}{4}$	1				$\sin \frac{\pi}{4} - 7 \cdot \sin \frac{\pi}{4}$	1
	$\sin -3 \cdot \frac{\pi}{2}$	1				$\sin \frac{\pi}{4} - \sin 7 \frac{\pi}{4}$	1
	$\sqrt{2} + \sqrt{2}$	1				$\sin \frac{\pi}{4} - \sin \frac{\pi}{4}$	1
						$\sin \frac{\pi}{4} - \sin 7 \cdot \frac{\pi}{4}$	1
						$\sin -6 \frac{\pi}{4}$	1
						$\sin 3 \cdot \pi$	1
						$\sin 6 \cdot \frac{\pi}{4}$	1
						$\sin \frac{\pi}{4} - 7 \cdot \frac{\pi}{4}$	1
						$\sqrt{2} - 2 \cdot \sqrt{2}$	1
						$\sqrt{2} - \sqrt{2}$	1
						x	1
Y.	42	128	36	129		51	126

Taulukko A.2 *Tehtävästä 3 poimitut opiskelijoiden vastaukset*

B. MATLABIN HERMOVERKOSTON TESTUSKOODI

```
% Avaa taulukon csv-tiedostosta, jossa on annetut vastaukset ja
% ennalta annetut pisteet eroteltuna omissa sarakkeissa "Vastaus"
% ja "Piste".
filename = "vastauset.csv";
data = readtable(filename,'TextType','string');

% Poistetaan tyhjat vastaukset ja kategorisoidaan pisteet
idxEmpty = strlength(data.Vastaus) == 0;
data(idxEmpty,:) = [];
data.Piste = categorical(data.Piste);

% Jaetaan vastaukset koulutus- ja testausosiin, suhteessa 4:1,
% ja jaetaan jaetut osat teksteihin ja pisteisiin
cvp = cvpartition(data.Piste,'Holdout',0.20);
dataTrain = data(training(cvp),:);
dataTest = data(test(cvp),:);

textDataTrain = dataTrain.Vastaus;
textDataTest = dataTest.Vastaus;
YTrain = dataTrain.Piste;
YTest = dataTest.Piste;

% Siivotaan ja vektoroidaan koulutusosan tekstit
textDataTrain = erasePunctuation(textDataTrain);
textDataTrain = lower(textDataTrain);
documentsTrain = tokenizedDocument(textDataTrain);

% Luodaan vektori-istutus
embeddingDimension = 100;
embeddingEpochs = 50;
```

```
emb = trainWordEmbedding(documentsTrain, ...
    'Dimension',embeddingDimension, ...
    'NumEpochs',embeddingEpochs, ...
    'Verbose',0)

% Optionaalinen vektoriavaruuden kuvaus
words = emb.Vocabulary;
V = word2vec(emb,words);
XY = tsne(V);
figure(3)
textscatter(XY,words)

% Luodaan teksteista viela verkostolle sopivat vektorit matriisiin,
% hyodyntaen luotua istutusta
sequenceLength = 100;
documentsTruncatedTrain = docfun(@(words)...
    words(1:min(sequenceLength,end)),documentsTrain);
XTrain = doc2sequence(emb,documentsTruncatedTrain);
for i = 1:numel(XTrain)
    XTrain{i} = leftPad(XTrain{i},sequenceLength);
end

% Hermoverkoston alustus
inputSize = embeddingDimension;
outputSize = 180;
numClasses = numel(categories(YTrain));

layers = [ ...
    sequenceInputLayer(inputSize)
    lstmLayer(outputSize,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer]

options = trainingOptions('adam', ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.01, ...
    'Plots','training-progress', ...
```

```
    'Verbose',0);

% Hermoverkoston koulutus
net = trainNetwork(XTrain,YTrain,layers,options);

% Kasitellaan testausosa samoin kuin koulutusosa kasiteltiin, uutta
% istutusta ei tehda, vaan kaytetaan koulutusosan istutusta
textDataTest = erasePunctuation(textDataTest);
textDataTest = lower(textDataTest);
documentsTest = tokenizedDocument(textDataTest);
documentsTruncatedTest = docfun(@(words)...
    words(1:min(sequenceLength,end)),documentsTest);
XTest = doc2sequence(emb,documentsTruncatedTest);
for i=1:numel(XTest)
    XTest{i} = leftPad(XTest{i},sequenceLength);
end

% Lasketaan verkon tarkkuus identtisten arviointien antamisessa
YPred = classify(net,XTest);
accuracy = sum(YPred == YTest)/numel(YPred)

% Edella kaytetyt funktiot sanojen vektorointiin istutuksen mukaan,
% ja lyhyiden vektorien pidentaminen matriisiin sovitettavaksi
function C = doc2sequence(emb,documents)
    parfor i = 1:numel(documents)
        words = string(documents(i));
        idx = ~ismember(emb,words);
        words(idx) = [];
        C{i} = word2vec(emb,words)';
    end
end

function MPadded = leftPad(M,N)
    [dimension,sequenceLength] = size(M);
    paddingLength = N-sequenceLength;
    MPadded = [zeros(dimension,paddingLength) M];
end
```

C. MATLABIN LSA-MALLIN TESTAUSKOODI

```
% Avaa taulukon csv-tiedostosta, jossa on annetut vastaukset ja
% ennalta annetut pisteet eroteltuna omissa sarakkeissa "Vastaus"
% ja "Piste".
filename = "vastaukset.csv"; %
data = readtable(filename,'TextType','string');

% Poistetaan tyhjat vastaukset
idxEmpty = strlength(data.Vastaus) == 0; %
data(idxEmpty,:) = [];

% Jaetaan vastaukset koulutus- ja testausosiin, suhteessa 4:1,
% ja jaetaan jaetut osat teksteihin ja pisteisiin
cvp = cvpartition(data.Piste,'Holdout',0.2); %
dataTrain = data(training(cvp),:);
dataTest = data(test(cvp),:);

textDataTrain = dataTrain.Vastaus; %
textDataTest = dataTest.Vastaus; %
YTrain = dataTrain.Piste; %
YTest = dataTest.Piste; %

% Siivotaan, vektoroidaan koulutusosan tekstit ja luodaan sanasakki
textDataTrain = erasePunctuation(textDataTrain);
textDataTrain = lower(textDataTrain);

documentsTrain = tokenizedDocument(textDataTrain);

bag = bagOfWords(documentsTrain);

% Vektoroidaan myos testausosan tekstit
documentsTest = tokenizedDocument(textDataTest);
```

```
% Alustetaan muuttujia:
% LSA mallin antamat arvosanat
result = zeros(size(YTest,1),1);
% painotettu tarkkuus, jossa yhden pisteen eron arvo on 1/2
partialaccuracy = 0;
% tarkkuus identtisille arvosanoille
accuracy = 0;
% parhaimman tuloksen saavuttanut LSA-avaruuden ulottuvuuksien
% maara
bestdim = 0;
% parhaimman tuloksen saavuttanut k-lahimman naapurin k
bestk = 0;
% parhain tarkkuus
bestaccuracy = 0;
% parain painotettu tarkkuus
bestapartialaccuracy = 0;
% parhaimman tuloksen erot
bestcompare = [];

% kokeillaan 5-80 ulottuvuuksilla askelvalilla 5
for dim = 5:5:min(80,size(textDataTrain,1))

    % luodaan LSA-malli koulutussanasakista
    mdl = fitlsa(bag,dim);
    % poimitaan mallista vektorien arvot
    dscores = mdl.DocumentScores;
    % sovitetaan testattavat vektorit LSA_malliin
    dscoresTest = transform(mdl,documentsTest);

    % haetaan k-lahinta naapuria kun k on 1-9, askelvalilla 2 jotta
    % keskiarvon laskeminen on yksiselitteinen
    for k = 1:2:9

        % haetaan jokaisen testattavan vektorin k-lahin naapuri, ja
        % niiden valmiiden arvosanojen keskiarvo
        for i = 1:size(dscoresTest,1)
            kIdx = knnsearch(dscores,dscoresTest(i,:),'K',k);
            result(i) = median(YTrain(kIdx,1));
        end
```

```
        % lasketaan tarkkuudet
        accuracy = sum(YTest==result)/(numel(YTest));
        partialaccuracy = sum(2-abs(YTest-result))/(numel(YTest)*2);

        % tarkastetaan onko saavutettu parempi tarkkuus
        if accuracy > bestaaccuracy
            bestaccuracy = accuracy;
            bestapartialaccuracy = partialaccuracy;
            bestk = k;
            bestdim = dim;
            bestcompare = [YTest,result];
        end
    end
end
```